### QuickSim II Training Workbook

Software Version 8.5\_1



Copyright © 1991 - 1995 Mentor Graphics Corporation. All rights reserved. Confidential. May be photocopied by licensed customers of Mentor Graphics for internal business purposes only. The software programs described in this document are confidential and proprietary products of Mentor Graphics Corporation (Mentor Graphics) or its licensors. No part of this document may be photocopied, reproduced or translated, or transferred, disclosed or otherwise provided to third parties, without the prior written consent of Mentor Graphics.

The document is for informational and instructional purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in the written contracts between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**RESTRICTED RIGHTS LEGEND** Use, duplication, or disclosure by the Government is subject to restrictions as set forth in the subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

A complete list of trademark names appears in a separate "Trademark Information" document.

Mentor Graphics Corporation 8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.

This is an unpublished work of Mentor Graphics Corporation.

#### TABLE OF CONTENTS

Purpose of Course	About This Training Workbook xiii	
Course Overview	Purpose of Course	xiv
Workbook Format       xvi         Lab Exercises       xvii         Module 1       1-1         Overview of QuickSim II       1-1         Overview_       1-2         Lesson       1-3         Topic Discussion Format 1-4       1-3         QuickSim II in Mentor Graphics Tool Set 1-6       1-6         The QuickSim II Simulation Process 1-8       1-10         QuickSim II Session Window 1-12       2         QuickSim II Application Windows 1-14       2         Loading/Connecting Waveforms 1-16       1-1         Running the Simulation 1-20       3         Analyze, Modify, and Save Results 1-22       2         Exiting QuickSim II 1-24       2         Lab Command Options 1-26       2         Using Strokes 1-28       1-32         Lab Preview 1-30       1-32         Procedure 1: Copying the Training Data       1-33         Procedure 2: QuickSim II       1-37         Test Your Knowledge       1-47	Course Overview	XV
Lab Exercises       xvii         Module 1       1-1         Overview of QuickSim II       1-1         Overview	Workbook Format	xvi
Module 1       1-1         Overview of QuickSim II       1-1         Overview	Lab Exercises	xvii
Overview of QuickSim II       1-1         Overview       1-2         Lesson       1-3         Topic Discussion Format 1-4       1-3         QuickSim II in Mentor Graphics Tool Set 1-6       1-6         The QuickSim II Simulation Process 1-8       1-0         QuickSim II Session Window 1-12       1-2         QuickSim II Application Windows 1-14       1-4         Loading/Connecting Waveforms 1-16       1-1         Running the Simulator 1-18       1-2         Stopping the Simulation 1-20       1-32         Analyze, Modify, and Save Results 1-22       1-2         Exiting QuickSim II 1-24       1-3         Lab Command Options 1-26       1-32         Vising Strokes 1-28       1-32         Performing Lab Procedures       1-32         Procedure 1: Copying the Training Data       1-33         Procedure 2: QuickSim II       1-37         Test Your Knowledge       1-46         Module Summary       1-47	Module 1	
Overview	Overview of QuickSim II	1-1
Lesson	Overview	1-2
Topic Discussion Format 1-4QuickSim II in Mentor Graphics Tool Set 1-6The QuickSim II Simulation Process 1-8Invoking from the Design Manager 1-10QuickSim II Session Window 1-12QuickSim II Application Windows 1-14Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulator 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab ExercisePerforming Lab Procedures1-32Procedure 1: Copying the Training Data1-37Test Your Knowledge1-47	Lesson	1-3
QuickSim II in Mentor Graphics Tool Set 1-6The QuickSim II Simulation Process 1-8Invoking from the Design Manager 1-10QuickSim II Session Window 1-12QuickSim II Application Windows 1-14Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulator 1-18Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab ExercisePerforming Lab Procedures1-32Procedure 1: Copying the Training Data1-37Test Your Knowledge1-47	Topic Discussion Format 1-4	
The QuickSim II Simulation Process 1-8Invoking from the Design Manager 1-10QuickSim II Session Window 1-12QuickSim II Application Windows 1-14Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab ExercisePerforming Lab ProceduresProcedure 1: Copying the Training DataProcedure 2: QuickSim II1-37Test Your KnowledgeModule Summary	QuickSim II in Mentor Graphics Tool Set 1-6	
Invoking from the Design Manager 1-10QuickSim II Session Window 1-12QuickSim II Application Windows 1-14Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	The QuickSim II Simulation Process 1-8	
QuickSim II Session Window 1-12QuickSim II Application Windows 1-14Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulator 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	Invoking from the Design Manager 1-10	
QuickSim II Application Windows 1-14Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	QuickSim II Session Window 1-12	
Loading/Connecting Waveforms 1-16Running the Simulator 1-18Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	QuickSim II Application Windows 1-14	
Running the Simulator 1-18Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	Loading/Connecting Waveforms 1-16	
Stopping the Simulation 1-20Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	Running the Simulator 1-18	
Analyze, Modify, and Save Results 1-22Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	Stopping the Simulation 1-20	
Exiting QuickSim II 1-24Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	Analyze, Modify, and Save Results 1-22	
Lab Command Options 1-26Using Strokes 1-28Lab Preview 1-30Lab Exercise	Exiting QuickSim II 1-24	
Using Strokes 1-28 Lab Preview 1-30 Lab Exercise	Lab Command Options 1-26	
Lab Preview 1-30Lab Exercise	Using Strokes 1-28	
Lab Exercise1-32Performing Lab Procedures1-32Procedure 1: Copying the Training Data1-33Procedure 2: QuickSim II1-37Test Your Knowledge1-46Module Summary1-47	Lab Preview 1-30	
Performing Lab Procedures1-32Procedure 1: Copying the Training Data1-33Procedure 2: QuickSim II1-37Test Your Knowledge1-46Module Summary1-47	Lab Exercise	1-32
Procedure 1: Copying the Training Data1-33      Procedure 2: QuickSim II1-37      Test Your Knowledge1-46      Module Summary1-47	Performing Lab Procedures	1-32
Procedure 2: QuickSim II1-37 Test Your Knowledge1-46 Module Summary1-47	Procedure 1: Copying the Training Data	1-33
Test Your Knowledge1-46Module Summary1-47	Procedure 2: QuickSim II	1-37
Module Summary1-47	Test Your Knowledge	1-46
	Module Summary	1-47

Module 2	
Waveform Database Concepts	2-1
Overview	2-2
Lesson	2-3
QuickSim II Architecture 2-4	
QuickSim II Design Flow 2-6	
Cross-Window Selection 2-8	
Waveform Databases 2-10	
Predefined Waveform Databases 2-12	
Default Waveform Databases 2-14	
Waveform Database Architecture 2-16	
Waveforms 2-18	
Basic 12-State Logic Events 2-20	
Other Logic Event Types 2-22	
Waveform Storage Formats 2-24	
Loading Waveform Databases 2-26	
Saving Waveform Databases 2-28	
Unloading Waveform Databases 2-30	
Lab Preview 2-32	
Lab Exercise	2-34
Procedure 1: Loading Waveform Databases	2-34
Procedure 2: Waveforms vs. Pin and Net Activity	2-43
Procedure 3: Saving Waveform Databases	2-45
Test Your Knowledge	2-47
Module Summary	2-48
Module 3	
QuickSim II Design Invocation	3-1
Overview	3-2
Lesson	3-3
Design Manager Tool Invocation 3-4	
Invoking QuickSim II from the Shell 3-6	
Electronic Designs 3-8	
Design Viewpoints and QuickSim II 3-10	

Design Viewpoint Creation 3-12	
DVE Invocation 3-14	
DVE Default Windows 3-16	
Setting Up a Viewpoint for QuickSim II 3-18	
Design Configuration Window 3-20	
Resolving Property Values 3-22	
DVE Design Checks 3-24	
Using TimeBase 3-26	
Reporting Timing Information 3-28	
Lab Preview 3-30	
Lab Exercise	3-32
Procedure 1: Creating a Viewpoint	3-32
Procedure 2: Invoking QuickSim II	3-40
Test Your Knowledge	3-46
Module Summary	3-47
Module 4	4.1
Simulation Models	4-1
Overview	4-2
Lesson	4-3
Database Structure 4-4	
Model Selection 4-6	
Component Interface Browser (CIB) 4-8	
"Browsing" a Component Interface 4-10	
Using the Component Window 4-12	
Modeling Methods 4-14	
Modeling Comparison 4-16	
Schematic Models 4-18	
Other Model Examples 4-20	
Memory Table Models 4-22	
Initializing RAMs and ROMs 4-24	
How Models Are Registered 4-26	
Verifying Models 4-28	
Technology Files 4-30	
Lab Preview 4-32	

Lab Exercise	4-34
Procedure 1: Examining Models	4-34
Procedure 2: Verifying Models	4-44
Procedure 3: Changing a Technology File	4-48
Test Your Knowledge	4-52
Module Summary	4-53
Module 5	
Setup and Expressions	5-1
Overview	5-2
Lesson	5-3
Simulation Setup 5-4	
Setting Up the Session (review) 5-6	
Setting Up the SimView Interface 5-8	
Setting Up the Environment 5-10	
Setting Up the Kernel 5-12	
Reporting Setup Conditions 5-14	
Saving Setup Conditions 5-16	
Restoring Setup Conditions 5-18	
QuickSim II Startup Files 5-20	
Keeping Circuit Activity 5-22	
Simulation Timing Modes 5-24	
Scaling Delays 5-26	
Simulation Expressions 5-28	
Expression Examples 5-30	
More Expression Examples 5-32	
Lab Preview 5-34	
Lab Exercise	5-36
Procedure 1: Setting Up QuickSim II	5-36
Procedure 2: Restoring the SimView Setup	5-44
Procedure 3: Creating and Using Expressions	5-46
Test Your Knowledge	5-52
Module Summary	5-53

Module 6	
Design Model Initialization	6-1
Overview	6-2
Lesson	6-3
QuickSim II Logic States 6-4	
Multiple Driver Resolution 6-6	
QuickSim II Initialization Process 6-8	
Default vs. Classic Initialization 6-10	
The Initializing Process 6-12	
Change Warning Start 6-14	
Resetting the Simulator to Time Zero 6-16	
Window Gadgets (setup) 6-18	
Cross-Window Selection 6-20	
Source Viewing 6-22	
Using Selection Filters 6-24	
Lab Preview 6-26	
Lab Exercise	6-28
Procedure 1: Initializing Signal States	6-28
Test Your Knowledge	6-35
Module Summary	6-36
Module 7	
Creating and Modifying Stimulus	7-1
Overview	7-2
Lesson	7-3
Stimulus Management 7-4	
Applying Stimulus as Events 7-6	
The Force Command 7-8	
The Stimulus Palette 7-10	
Issuing Forces 7-12	
Issuing Clock Forces 7-14	
Stimulus Pattern Generation 7-16	
Creating a Waveform Databases 7-18	
Creating a Waveform 7-20	

Waveform Editing 7-22	
Setup Waveform Editor 7-24	
Selecting Waveforms to Edit 7-26	
Lab Preview 7-28	
Lab Exercise	7-30
Procedure 1: Creating Waveforms	7-30
Procedure 2: Working with Waveform Databases	7-39
Test Your Knowledge	7-43
Module Summary	7-44
Module 8	
Running a Simulation	8-1
Overview	8-2
Lesson	8-3
Performing a QuickSim II Run 8-4	
QuickSim II Analysis Evaluations 8-6	
Timing Wheel Concept 8-8	
Iterations 8-10	
Timing Accuracy 8-12	
Setting Run Parameters 8-14	
Using Breakpoints 8-16	
Using Actionpoints 8-18	
Saving and Restoring Simulation States 8-20	
Batch Simulations 8-22	
Setting Up Batch Simulations 8-24	
Running a Batch Simulation 8-26	
Lab Preview 8-28	
Lab Exercise	8-30
Procedure 1: Evaluations and Oscillations	8-30
Procedure 2: Resolution and timing	8-38
Test Your Knowledge	8-42
Module Summary	8-43

Module 9	
Examining Results and Waveforms	9-1
Overview	9-2
Lesson	9-3
Results Analysis in the Design Flow 9-4	
Examining Results (SimView) 9-6	
Palettes Provide Access 9-8	
The Trace Window 9-10	
Trace Window Cursors 9-12	
The List Window 9-14	
The Monitor Window 9-16	
Using Monitor Flags 9-18	
Waveform Edge Deltas 9-20	
Viewing a Specific Time 9-22	
Bus Structures 9-24	
Defining Buses (setup) 9-26	
Assertions 9-28	
Assertion Tests 9-30	
Converting Waveforms to Assertions 9-32	
Lab Preview 9-34	
Lab Exercise	9-36
Procedure 1: Analysis Using SimView	9-36
Procedure 2: Analysis Using Assertion Tests	9-45
Test Your Knowledge	9-50
Module Summary	9-51
Module 10	
Simulating Hierarchical Designs	10-1
Overview	10-2
Lesson	10-3
QuickSim II Features 10-4	
Debug Hierarchy Palette 10-6	
Examining Design Hierarchy 10-8	
Latching Design Version 10-10	

Changing Properties 10-12	
Changing Selected Property Values 10-14	
Selecting by Property 10-16	
Changing Named Property Values 10-18	
Changing a Model 10-20	
Design Changes in QuickSim II 10-22	
General Effects of Design Changes 10-24	
Non-Connectivity Changes 10-26	
Connectivity Changes 10-28	
Back Annotation 10-30	
Back Annotation Object Priority 10-32	
Lab Preview 10-34	
Lab Exercise	10-36
Procedure 1: Making Design Changes	10-36
Test Your Knowledge	10-45
Module Summary	10-46
Modulo 11	
Debugging Design Analysis	11-1
Overview	11-2
Lesson	11-3
Debug Gates Palette 11-4	
Using Probes 11-6	
Locating Unknown (X) Sources 11-8	
Design Changes Palette (SimView) 11-10	
Design Timing Constraints 11-12	
Setting Instance Constraint Mode 11-14	
Spike Models 11-16	
X-Immediate Spike Model 11-18	
Suppress Spike Model 11-20	
Technology File Spike Model 11-22	
Changing Instance Contention Models 11-24	
Conditions That Affect Kernel Setup 11-26	
Lab Preview 11-28	
Lab Exercise	11-30

Procedure 1: Setting Up the Debug Environment	11-30
Procedure 2: Fixing Constraint Violations	11-36
Test Your Knowledge	11-43
Module Summary	11-44
Appendix A	
Solutions	A-1
Module 1 Solutions	A-1
Lab Answers	A-1
Test Your Knowledge Solutions	A-1
Module 2 Solutions	A-3
Lab Answers	A-3
Test Your Knowledge Solutions	A-4
Module 3 Solutions	A-5
Lab Answers	A-5
Test Your Knowledge Solutions	A-5
Module 4 Solutions	A-7
Lab Answers	A-7
Test Your Knowledge Solutions	A-7
Module 5 Solutions	A-9
Lab Answers	A-9
Test Your Knowledge Solutions	A-10
Module 6 Solutions	A-11
Lab Answers	A-11
Test Your Knowledge Solutions	A-11
Module 7 Solutions	A-13
Lab Answers	A-13
Test Your Knowledge Solutions	A-13
Module 8 Solutions	A-15
Lab Answers	A-15
Test Your Knowledge Solutions	A-15
Module 9 Solutions	A-17
Lab Answers	A-17
Test Your Knowledge Solutions	A-17
Module 10 Solutions	A-18

Lab Answers	A-18
Test Your Knowledge Solutions	A-18
Module 11 Solutions	A-19
Lab Answers	A-19
Test Your Knowledge Solutions	A-19
Appendix B	
Design Manager Icons	B-1
Data Object Icons	B-1
Idea Station Tool Icons	B-4

# **About This Training Workbook**

Welcome to the *QuickSim II Training Workbook*. In this course you will learn about QuickSim II, the Mentor Graphics digital logic simulator. You will also be introduced to digital modeling methods and utilities that support QuickSim II.



This training workbook requires that you know how to use the common user interface of the Falcon Framework. QuickSim II uses this interface for the window, mouse, and keyboard environment. For more information about Falcon Framework, refer to *Getting Started with Falcon Framework*.

If you are using this document online in INFORM, you will see occasional highlighted text. On a black and white display, this text appears enclosed in a rectangle, and on a color display using the default color map, the text is blue. The highlighted text is a hypertext link to related materials in this and other documents. If you click the Select mouse button on a hypertext link, the linked location will be displayed.



For information about the documentation conventions used in this manual, refer to *Mentor Graphics Corporation Documentation Conventions*.

### **Purpose of Course**

This workbook presents QuickSim II, the Mentor Graphics digital logic simulator. The purpose of the training workbook is to:

- Show how to use the common simulation (SimView) user interface.
- Familiarize you with the Design Viewpoint Editor (DVE) and the viewpoint concept. You will get hands-on experience using DVE in the lab.
- Show how to create stimulus using manual and batch methods, and how to manage stimulus and results in waveform database files.
- Present digital logic simulation concepts.
- Present command line and Design Manager invocation of QuickSim II.
- Familiarize you with the QuickSim II user interface. You will learn how to complete a digital simulation using the QuickSim II user interface.
- Present the V8 digital modeling methods. This includes sheet-based models, QuickPart models, behavioral language models (BLMs), VHDL models, technology files, and model selection. Note: VHDL model creation is not covered in this training workbook.
- Present the concepts of back annotation. Design back annotation information is versioned, and is kept with the design viewpoint.

This workbook *does not* address the following:

- Falcon Framework utilities, such as Design Manager, IDM (Integrated Design Manager), Notepad editor, or DSS. Falcon Framework training is available in *Getting Started with the Falcon Framework*.
- How to use Design Architect. Schematic capture is not taught in this training workbook, but may be required in some of the lab exercises. Design Architect training is presented in the *Design Architect Training Workbook*.

### **Course Overview**

This workbook is divided into eleven modules. Here is a brief description of each:

- 1. **Overview of QuickSim II**. Includes information on invoking, running and stopping a simulation, and exiting QuickSim II, saving results.
- 2. **Waveform Database Concepts**. Describes general simulation concepts, including the simulation process and waveform database concepts.
- 3. **QuickSim II Design Invocation**. Explains Design Manager and command line invocation. Describes what happens during invocation.
- 4. **Simulation Models**. Includes modeling methods such as technology files, QuickPart tables, initializing memory, and the component interface.
- 5. **Setup and Expressions**. This module examines expressions, saving and restoring setup conditions, and setting up the user environment
- 6. **Design Model Initialization**. Discusses default and classic initialization. You will use the initialize command and fix initialization problems.
- 7. Creating and Modifying Stimulus. Discusses stimulus management. You will generate waveforms using forces, stimulus generator, and AMPLE.
- 8. **Running a Simulation.** Discusses how the kernel performs and analysis, circuit initialization, logic states, kernel setup, and performing the run.
- 9. **Examining Results and Waveforms**. You will connect a results waveform database into SimView and use expressions and assertions to compare data.
- 10. **Simulating Hierarchical Designs.** You will simulate lower levels of design hierarchy, then use those results in an upper level hierarchical simulation.
- 11. **Debugging Design Analysis**. Discusses debug palettes, and probes. You will use breakpoints/actionpoints, create buses and troubleshoot unknowns

### Workbook Format

Within this workbook you will find:

- Table of Contents: A listing of section titles, figures and tables.
- About This Training Workbook: Contains general workbook information and explains how to use this student workbook.
- Modules: Each module has the following structure:
  - **Overview:** Description of the module contents and a list of objectives for using the material. The objectives describe what you should know or be able to do after completing the material.
  - Lesson: Narrative explanations of concepts and practice procedures.

Left- and right-facing pages of the lesson form a "topic." The left page contains brief descriptions, figures, and tables. The right page explains concepts, and contains document references.

• Lab Exercises: Complete materials to perform the hands-on lab session for each module.

Each lab session may include several **Lab Procedures** which are step-bystep lab instructions about a simulation concept.

- **Test Your Knowledge:** Questions testing your knowledge of the material presented in this module.
- Module Summary: A text review of what was learned in this module.
- Appendixes:
  - A: Solutions contain Lab Exercise and Test Your Knowledge answers.
  - **B:** Table of navigation and tool icons used with Idea Station objects.

### Lab Exercises

Several designs are used in this training workbook. A simple D-type flip flop, shown in Figure 2 on page xviii, is used in the first few labs so that design details do not complicate the learning process. The *card\_reader/add\_det* design, shown in Figure 1 below, is used in the final lab to allow you to explore more complex design hierarchies. Both designs were created in Design Architect, and are the same designs used in the *Design Architect Training Workbook*.



#### Figure 1. add\_det Lab Circuit

Each circuit shows a different aspect of the modeling and analysis process. You may even want to try some of the lab procedures on your own designs.



#### Figure 2. D-type Flip Flop Lab Circuit

Lab Exercises are divided into numbered steps that are short groups of actions that form an operation. Procedures may require several steps to complete. A step is divided into three parts:

- 1. What you will do. A short description of what you should expect to complete at the end of the step. The details of how to perform the actions are not given in this part.
- 2. **How you will do it**. A detailed description of the things you must do to complete the step. This includes caveats and helpful hints.
- 3. What is the result. Usually a picture or a brief explanation, showing the outcome of this step. You use this information to verify that you have done the step correctly. Remember that each step builds on the preceding step. If you incorrectly performed a previous step, the current step may be impossible to complete correctly.

When you have completed the lab exercise, you are asked to answer a set of "Test Your Knowledge" review questions. Some of these questions are based on knowledge you gained from the Lab Exercise. Solutions to these questions are contained in Appendix A.

# Module 1 Overview of QuickSim II

Overview	1-2
Lesson	1-3
Topic Discussion Format	1-4
QuickSim II in Mentor Graphics Tool Set	1-6
The QuickSim II Simulation Process	1-8
Invoking from the Design Manager	1-10
QuickSim II Session Window	1-12
QuickSim II Application Windows	1-14
Loading/Connecting Waveforms	1-16
Running the Simulator	1-18
Stopping the Simulation	1-20
Analyze, Modify, and Save Results	1-22
Exiting QuickSim II	1-24
Lab Command Options	1-26
Using Strokes	1-28
Lab Preview	1-30
Lab Exercise	1-32
Procedure 1: Copying the Training Data	1-33
Procedure 2: QuickSim II	1-37
Test Your Knowledge	1-46
Module Summary	1-47

#### **Overview**



### Lesson

On completion of this module, you should:

- Know the name of the applications used with QuickSim II for digital analysis.
- Be able to describe features of QuickSim II.
- Set up standard schematic view, Trace, List, and Monitor windows.
- Apply stimulus to a signal using single forces (lab only).
- Apply clock stimulus to a signal (lab only).
- Run a simple simulation.
- Display the QuickSim II transcript, and paste transcript information into an edit pad.



If you are unfamiliar with Falcon Framework, please go through *Getting Started with Falcon Framework* before starting this training workbook.



You should allow approximately 1 hour and 15 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

## **Topic Discussion Format**



QuickSim II Training Workbook, 8.5\_1 November 1995

## **Topic Discussion Format**

The lesson is divided into many topics that are presented in left page-right page format as shown in the figure above. The visual material is presented on the left page and explanatory text and references are on the right page.

Topic pages begin on the next page.

### **QuickSim II in Mentor Graphics Tool Set**



#### **Applications -- Process designs**

#### Models -- Design building blocks

### **QuickSim II in Mentor Graphics Tool Set**

The QuickSim II logic simulator is a computer program that allows you to test a 'software breadboard' of a digital hardware design. QuickSim II operates on a model of a digital logic circuit that consists of parts you have connected together with Design Architect, or created using the System-1076 modeling method.

The figure illustrates the role of QuickSim II in the Mentor Graphics tool set. Other Mentor Graphics applications that can be used to process your design are:

- AutoLogic. Synthesizes a schematic from VHDL code so that you can layout your design.
- **QuickPath.** A critical path analysis application that provides path length and slack (setup and hold) checking.
- **QuickGrade II.** A statistical fault grader for performing rapid and accurate fault analysis of your design.
- **QuickFault II.** A deterministic fault simulator that thoroughly examines your design to determine fault coverage prior to design test.
- ASIC Layout. This is a vendor specific process where ASIC vendor libraries can be used in Design Architect to produce ASIC designs.
- **Board Station.** Allows you to create package and board layout and fabrication information.

With QuickSim II, you can apply stimulus to the design, run the simulation, analyze the results, and then modify the design based on those results. You can then reset the simulator, optionally apply more stimulus to the design (the simulator maintains the original set of stimulus), and start the cycle over. When the design functions correctly, you can save the stimulus and results directly with the design, to promote consistent and reliable design management.

## **The QuickSim II Simulation Process**



### **The QuickSim II Simulation Process**

The figure shows steps that you might use in the typical QuickSim II simulation process. As shown in the figure, QuickSim II as well as Design Architect can be invoked from the Design Manager. The simulation process steps include:

- **Design Creation/Capture**. Use Design Architect to create the EDDM (Electronic Design Data Model) database along with graphical information.
- **Simulation**. Use QuickSim II to run various simulation iterations to verify the design's functionality, timing, and back annotations. A standard simulation run takes the following shape:
  - **Invoke QuickSim II on a design**. QuickSim II assumes that a design has been created and captured using Design Architect.
  - **Generate and refine stimulus**. Uses waveform databases; the waveform editor allows graphical modification of stimulus.
  - **Run simulation**. QuickSim II and the simulation kernel, is required to simulate a design.
  - **Stop simulation**. You can stop the simulations manually, or let QuickSim II stop automatically as the result of a breakpoint. You can also specify a finite run length.
  - **Analyze results**. SimView supports cross-window selection; multiple Trace/List windows.

### **Invoking from the Design Manager**

#### Issue: shell> \$MGC\_HOME/bin/dmgr



To invoke QuickSim II:

- Select design an choose: Open > QuickSimII
- (or) Double-click on QuickSimll tools icon

### **Invoking from the Design Manager**

The Design Manager is the preferred method to examine design objects and invoke applications. To bring up the Design Manager, in a shell issue:

\$MGC\_HOME/bin/dmgr

When the Design Manager invokes, there are two default windows created: a Tools window and a navigator window. The navigator objects are shown as icons with each icon representing the type of object and the application that created it. Two types of design navigation windows are available. You can create additional navigator windows from the Windows pulldown menu:

#### Windows > Open Navigator > View by Icon or

#### Windows > Open Navigator > View by Name

If you select the design object in a navigator window, and examine the **Open** > submenu items from the navigator popup menu, you are given a list of applications that you can invoke on the selected object. For example, to invoke QuickSim II on *design\_1*, first select *design\_1* in the navigator window, and then choose the following in the popup menu:

#### Open > <u>#</u> QuickSimII

Each item is numbered if you choose to use the keyboard method of menu selection. The number in the menu pick ( $\underline{\#}$ ) may vary depending on the number of applications supported by *design\_1*. A separate process will be spawned invoking QuickSim II on the selected design.

The Design Manager is also used to copy, move, verify and release designs.



For more information on the Design Manager, refer to the *Design Manager User's Manual*.

## **QuickSim II Session Window**



### **QuickSim II Session Window**

The windows that are presented, by default, are the session window, the palette area, and the softkeys area. In a subsequent module, you will learn how to configure the interface as part of invocation.

The menu bar shows default pulldown menu names that are available in the default environment. These menus contain the following:

**MGC**. User interface options common to all applications. It includes session setups, screen printing, transcripting, location map configuration, the Notepad editor, userware, and Integrated Design Management (iDM).

**File**. Opening the schematic view window, saving and restoring states/setups, printing, and checking the design rules.

**Edit**. Selection and unselection filters, property manipulation, latching versions, and clipboard functions.

**Add**. Adds signals to the Trace, List and Monitor windows, adds breakpoints, defines buses, adds probes, and keeps signals.

Delete. Deletes the same objects that were added in the Add menu.

**Setup**. Sets up the kernel, environment, window defaults, run parameters, and creates an empty Trace, List, or Monitor window. All stimulus creation is also contained in this menu.

Run. Various ways of running or resetting a simulation.

Report. Accesses the numerous reports available in QuickSim II.

View. Contains zooms, view all/area, scrolls, and viewing panels.

Help. Access application Quick help and online documentation Reference help.

Each menu name contains an underlined letter. If you press the Pulldown Menu function key, the underlined letter can be typed to access the menu, instead of using mouse access.

## **QuickSim II Application Windows**



### **QuickSim II Application Windows**

During an interactive QuickSim II simulation, you will create *client* windows that allow you to view schematics, stimulus, results, and conditions. A client window is created within a session window and provides a specific function.

The illustration on the previous page shows a QuickSim II session with a typical window environment displayed. These client windows are part of the SimView user interface and are described in SimView training. Here is a brief description of several standard client windows:

- Schematic view. This graphical window allows you to examine your design, and to select design objects. This window supports cross-window highlighting and source viewing. The title of the window displays the path from the design root to the displayed sheet.
- List window. This window lets you display signal and waveform information in textual form. It displays simulation times and state information for each signal that is added to the window. Contents of the window can be printed or written to a file.
- **Trace window**. This graphical window lets you examine signal and waveform states in graphical form. The time axis is horizontal, and signals are displayed vertically. Buses can be combined or traced as individual signals. Trace window cursors can be added to the window to aid in identifying states and times. A delta-time function determines the delay between state changes.
- **Monitor window**. This window gives the current simulation time. It also gives the current state of any signal listed, in textual form. This window supports cross-window highlighting.
- **Report windows** (not shown). These windows allow you to examine many of the conditions, warnings, files, setups, and other information related to a QuickSim II digital simulation.

## **Loading/Connecting Waveforms**

#### (Menu bar) > File > Load > Waveform DB



- Load waveforms from disk object into QuickSim II program memory
- Connect waveforms to design as stimulus

## **Loading/Connecting Waveforms**

Waveform databases must be created or loaded in program memory to be used. When you load a waveform database, it is copied from disk into program memory. Once copied into program memory, then the waveforms need to be connected to the design input nets as stimulus. In this way the design has data in which to process thereby creating results data. The results data is analyzed to ensure that the stimulus waveforms were processed correctly.

You can create and modify stimulus waveforms, and analyze results waveforms in preparation for a simulation session. You apply (connect) stimulus waveforms to a design object to drive the simulation and extract (keep) results waveforms from a design object.

## **Running the Simulator**




# **Running the Simulator**

You begin a simulation run by issuing the Run command, choosing one of the **Run > Simulation >** menu items, or using the **Run** palette button. If you use the Run command, you can specify a time, in user time units, for which you want the simulator to run. You can also specify whether that run time is absolute, or relative to current simulator time.

If no run time is specified, the run will continue until the simulator encounters a breakpoint, a Stop command, or information you might have supplied with the Setup Run command.

For example, if you want to run the simulator for 1000 time units relative to current simulator time, you can click on the palette item:

### [Palette] Run

Choose **For Time** from the menu that appears. When the prompt bar appears, enter 1000 in the "For Time" field, and click the **OK** button on the prompt bar.



For more information on running a simulation, refer to "Controlling Simulation" in the *SimView Common Simulation User's Manual*.

# **Stopping the Simulation**

Pausing the simulation:

- Breakpoint or pause operation. Leaves previous stop point.
- [Palette] Run > Resume button -- Continues running from a breakpoint or pause.

Stopping the simulation:

- Runs out of time
- Runs out of events (all quiet)
- The stop key:

	HP-Apollo	Sun	HP700
Stop	Ctrl-S	Ctrl-C	Ctrl-C
Suspend	Ctrl-X	Ctrl-S	Ctrl-S
Resume	Ctrl-C	Ctrl-Q	Ctrl-Q
Kill	Ctrl-Q	Ctrl-\	Ctrl-\

# **Stopping the Simulation**

If the simulator is running and you want to do more interactive work, you can suspend the simulation. For example, as the simulation is running, you see unknown signal states appearing in the list and trace window. By stopping the simulation immediately, you can examine the cause of these conditions.

If you have temporarily suspended a simulation by encountering a breakpoint, you can pick up where you left off by using the Resume Simulation command. The effect of the Resume Simulation command is to continue performing the simulation that was begun with the last Run command.

For example, to resume a paused simulation, use the following palette button:

### [Palette] Run > Resume

Stopping a simulation refers to the process of aborting the current simulation as soon as possible. Use this procedure when you've decided that further simulation of the design would serve no useful purpose. You might have seen via the Trace or List windows that the results of the simulation were very different from what you expected, and realize that you need to change the design before a useful simulation can be performed.

To abort the current simulation, press the appropriate Ctrl key as shown in the table on the previous page. For the HP/Apollo, this is the Ctrl-S key. For the Sun and HP700 workstations, this is Ctrl-C.

## Analyze, Modify, and Save Results

- Graphically analyze and modify waveform data using palette icons
- Rerun until design performs to specification



Note: This module does not detail results analysis. Refer to Module 9 for a detailed discussion.

# Analyze, Modify, and Save Results

Use the palettes to modify and analyze waveforms. More information about each of the palette icons is provided by choosing the **Help > On Palettes > Palette Descriptions** menu items.

The Stimulus palette provides icons that allow you to add and delete individual forces as well as clocks. This palette also provides icons to create, load, and save waveform databases.

The Waveform Editor palette allows you to graphically manipulate waveform events. You can add waveforms from the "force target" waveform database to a Trace window so they can be edited. Using the waveform editor, you can insert, copy, invert, or delete waveform transitions.

The Design Changes palette provides icons for changing a design while still in SimView. You can add, change, or delete design properties and then back annotate the changes into the design.

The Debug Gates palette provides several icons that assist you in analyzing a waveform. You can locate and backtrace to their source any unknown (X) values. You can create and move trace cursors that indicate the time and state of each waveform in the Trace window. Edge deltas can be displayed to assist you in determining delays and timing. You can also add Probes and Monitor windows which can be most useful in a SimView/UI-based simulation environment.

This module does not further discuss analysis techniques within a simulation, or the palettes mentioned above. All of these palettes and operations are discussed in detail in subsequent Modules.

# **Exiting QuickSim II**

### Choose: (Window Menu) > Close

Exit Qu	uickSim				
After Saving	Without saving				
Save Setup (WDBs, Windows, etc.)					
Save 'results Waveform DB					
OK Re	set Cancel				

Prompts you to save new or changed objects:

- Setup (SimView and kernel setup)
- Waveform Databases
  - 'results' Waveform Database
  - $\circ$  forces
  - user-created waveform databases

# **Exiting QuickSim II**

In general, there is no reason to exit the simulator unless you are ready to simulate another design or to log off the system. All other operations, including changing design connectivity, can be performed while QuickSim II is invoked.

When you do decide to exit the simulator, there are several ways you can go about it. You can use the manner acceptable in all Mentor Graphics applications by pulling down the menu that appears in the top-left corner of the session window (Window menu) and selecting the **Close** menu item. Or, you can issue the **Exit** command from within the simulator.

If you entered back annotations during the simulation and didn't subsequently save the viewpoint, a dialog box appears and gives you the chance to authorize the changes to be saved. Other simulator objects can be saved as well, such as the kernel states, QuickSim (kernel) setup, SimView (environment) setup, and all waveform databases. These objects will be discussed in more detail in a later module.

To exit without saving, pick the "Without saving" button and OK the dialog box. QuickSim II will execute a \$force\_exit function and terminate the session.

To save objects and then exit, pick "After Saving" item and click on all objects that you want to save, and then OK the dialog box. QuickSim II will save those objects you have chosen, and then exit.

To return to the simulation (no save or exit), click on the "Cancel" option.



For more information on exiting QuickSim II, refer to the *QuickSim II* User's Manual.

# **Lab Command Options**

'Best Way' Philosophy

• Menu Choices:

(Menu Bar) > Report > Setup > Stimulus

- Palette Buttons (click)
  - Palette Selection Button SETUP
  - Common Palette Button TRACE



• Application Commands & Functions



• Strokes

Select ADDR and Issue the \_\_\_\_\_\_ stroke

## Lab Command Options

For every operation you want to perform in QuickSim II, there are five or six different ways to issue each operation. It is not the intent of this training workbook to show you how to perform every operation using all of the different methods. The authors of this workbook have chosen a "best way" given the circumstances and feedback from experienced users.

However, you should know the methods available in QuickSim II to issue commands. This may help you choose your own "best way" based on your unique set of circumstances. The following list identifies the methods available, and gives you an example of how this method is presented in the lab exercises.

- Menu Choices. An instruction, such as Choose: (Menu bar) > Report > Setup > Stimulus indicates that you click the Select mouse button on "Report" in the menu bar and continue to click on named items in the cascade menu presented. By clicking on the menu item, rather than dragging (holding the Select mouse button), menus remain on the screen to examine.
- **Palette Buttons (click).** Three types of palette buttons exist. Palette selection buttons (red), Common palette buttons (blue), and Palette action buttons (black). You will see and instruction like "Click on the Open Sheet palette button".
- **Function (soft) Keys.** Rather than specify the physical key (Shift-F1) you will be instructed to use the logical key. Logical key names are mapped to physical location in the Softkeys area at the bottom of the QuickSim II window. For example: "Press the Unselect All key" means press the F2 key.
- Application Commands & Functions. Commands and functions are entered in the popup command line (just begin typing). Commands are not syntax critical, and allow minimum typing. Functions are entered exactly.
- **Strokes.** You can use a graphical shortcut for commands using the middle mouse button. Strokes are described on the next page.

# **Using Strokes**

		Quick Help on Stroke	s			
		Common SimView Stro	kes			
•	Activate Window 5	Unselect All 1478963	Vie 159	w Area		
•	View Centered Double Click MMB	Report Selected 1474123	Vie 951	w All		
▲	Pop Window 98741	Set Select Filter 32147	× Zoc 357	om In (2)		
	Select Window 1475963	Clear Select Filter	→ Zoc → 753	om Out (2)		
	Add Traces 96321	Delete 741236987	→ Op	en Selected 163		
	Add Lists 14789	→ Change	↓ Op 258	en Down Selected		
	321456987 ematic View Strokes	Move 74159	↓ Op ■ 852	en Up		
	Select Area 74123	Copy → 3214789 Copy → 3214789 Copy → 36987		en Sheet 187		
Ŧ	Open Down Nearest 258					
Str	roke Recognition Grid	More help on strokes				
		More Help on SimView Stro	kas			
	Other Strekes	Dialog Strokes Palette	Strokes	Report Strokes		
	Execute Last Menu 12369	■→ Execute 456	Scroll Up 53	■→ Close Windo 456	w	
∎-•	Execute prompt bar 456	← Cancel × S 654	Scroll Down		w	
_	Cancel prompt bar Stroke Recognition Grid					
	1 2 3 Strokes are drawn with the middle mouse key. They are recognized by					
	+ Help on Strokes 4 5 6 fitting the stroke path onto a 3x3 grid creating a numerical sequence					
		7 8 9				
		Print Ref Help C	lose			
					_	

# **Using Strokes**

Once you get the hang of using strokes, they are a quick and easy way to issue many commands and operations.

An attempt has been made to map the graphical stroke to resemble the first letter of the operation. For example "U" for Unselect All, "C" for copy, and "D" for delete. This makes many strokes easy to remember.

The figure on the opposite page shows the dialog boxes that give you help on strokes. To view this information, choose the (Menu bar) > Help > On Strokes menu item. The dialog box in the lower part of the figure is shown when you click on the "More Help on Strokes" button in the main Quick Help on Strokes dialog box.

Strokes that you may find useful are:

- Close Window
- Zoom In / View All
- Unselect All

# Lab Preview

- Set up lab design data using the Design Manager
- Invoke QuickSim II on a design using the Design Manager
- Issue stimulus to the design using a script
- Run the simulation
- Examine results
- Exit QuickSim II

## Lab Preview

In the lab exercise for this module, you will:

- 1. Invoke QuickSim II on a design using the Design Manager by selecting the design and choosing the **Open > quicksimII** menu item.
- 2. Execute a script to run a short simulation on the LATCH design. This script creates the schematic view window and the List window.
- 3. Run a simulation using the Run palette button.
- 4. Create the Trace window, adding the signals that are in the List window.
- 5. Exit QuickSim II without saving results.

## Lab Exercise



If you are reading this workbook online, you might want to print the lab exercises to have them handy when you are at your workstation.

### **Performing Lab Procedures**

Lab Exercises are divided into numbered steps that are short groups of actions forming an operation. Procedures may require several steps to complete. A step is divided into three parts:

- 1. What you will do. A short description of what you should expect to complete at the end of the step. The details of how to perform the actions are not given in this part.
- 2. How you will do it. A detailed description of the things you must do to complete the step. This includes caveats and helpful hints.
- 3. What is the result. Usually a picture or a brief explanation, showing the outcome of this step. You use this information to verify that you have done the step correctly. Remember that each step builds on the preceding step. If you incorrectly performed a previous step, the current step may be impossible to complete correctly.

When you have completed the lab exercise, you are asked to answer a set of "Test Your Knowledge" review questions. Some of these questions are based on knowledge you gained from the Lab Exercise. Solutions to these questions are contained in Appendix A.



For information on user interface objects shown in the lab exercises, refer to page 1-12 and page 1-26.

### **Procedure 1: Copying the Training Data**

In this procedure you will make a working copy of the training data for use with subsequent lab exercises. The following illustrates how you will make the copy:



This lab procedure requires that the MGC tree contain the training package "qsimnwp" as source for the training data you will copy. The path to this object is: *\$MGC\_HOME/pkgs/qsimnwp*. If this object does not exist, you (or your site administrator) need to install this training package before proceeding. The procedure for installing training packages is contained in the workstation-specific MGC software installation manual.

The rest of this procedure assumes that this training package has been properly installed.



- 1. Log into your workstation using your account and password.
- 2. If the Design Manager is not invoked, invoke it now in a shell as follows: shell> \$MGC\_HOME/bin/dmgr

The Tools window and the iconic navigator window should appear. If your configuration does not automatically create the iconic navigator window, create it now using the **Windows > Open Navigator > View by Icon** pulldown menu item.

- 3. Set up a training directory in your account (or the location designated by your instructor or system administrator) by doing the following steps:
  - a. First check to see if your student training directory exists in your home account (\$HOME). Use the Design Manager to navigate to the location of your training directory.
  - b. If it exists, go to Step 4. Otherwise, create the training directory using the menu item: (Menu bar) > <u>A</u>dd > Directory:
  - c. When the prompt bar appears, enter the path where you want this training directory to be located (normally under your \$HOME directory, or optionally in the /tmp directory):

ADD DI	Directory Pathname	\$HOME/training	OK	Cancel
			J	

Note that \$HOME appears in the pathname. This represents the path to the directory which houses your training directory. If you did not place your training directory beneath \$HOME, wherever you see "\$HOME" you should substitute your actual path for this name.

d. Click on the **OK** button.

- 4. Locate the *qsim851nwp* lab software for this training workbook, as follows:
  - a. From the navigator, click on the "Go To" icon at the bottom of the navigator window. The "Change directory to:" dialog box appears.
  - b. Enter the pathname to the MGC training source directory as shown:

Change directory to:
\$MGC_HOME/shared/training
OK Reset Cancel

c. Click on the **OK** button.

You should see a linked container named qsim851nwp



You will copy the linked object (and its contents) to your training directory.



If this linked container does not exist, you (or your site administrator) need to install the QuickSim II (qsimnwp) training package before proceeding. The procedure for installing training packages is contained in the workstation-specific MGC software installation manual.

- 5. Make a copy of the *qsim851nwp* object in your local training directory, naming it *qsim\_n* using the following steps:
  - a. Select the *qsim851nwp* object in the Navigator window using the Select mouse button.

The object highlights



b. Using the Navigator popup menu, choose:

```
(Navigator) > Edit > Copy
```

The COPy OBject prompt bar appears.

c. Enter the pathname in the Destination field to where you want the copy to be created, as shown:

СОР ОВ	Destination	\$HOME/training/qsim_n	Options	ОК	Can cel	

If you did not create the training directory in \$HOME, be sure to substitute the path to the directory in which you created your training directory for \$HOME in the above pathname.

d. Click on the **OK** button.

The "Working...." message appears in the message area. By specifying a destination object that does not exist (*qsim\_n*), the copy will be made using this name instead of the original name (*qsim851nwp*). The qsim\_n name is easier to type during lab sessions.

The following message appears when the copy operation is complete:

**İ** The Copy operation was successful

You are now ready to advance to Procedure 2.

### Procedure 2: QuickSim II

For this procedure, you should be logged in and have the Design Manager invoked in a shell, with a iconic navigator window available.

- 1. Using the navigator window, find and select the *qsim\_n/LATCH* component in your training directory as follows:
  - a. Click on the "Go To" button at the bottom of the Design Manager Navigator window.
  - b. Enter the following path in the prompt bar destination field.



You should see the following icons in the Navigator window: "LATCH", "MEMORY", "OSC", "add\_convert", "add\_det", "lib", and "scripts".

			training/qsim	_n		
*****						
	LATCH	MEMORY	OSC add_convert	add_det	lib	scripts

c. Select the LATCH icon.



This icon represents the design object LATCH. The icon highlights.

2. Open QuickSim II on the LATCH design object by choosing:

#### (Navigator popup menu) > Open > QuickSimII

As QuickSim II invokes, the SimView/UI loads first. SimView/UI is the common simulation user interface. It provides most of the selecting, viewing, and common simulation commands. When the QuickSim II kernel is attached, the window is labeled QuickSim II as shown on page 1-12.

- 3. Grow the QuickSim II window to fill the entire screen by clicking the mouse button on the session window Maximize button . (Note: some window environments may have a menu choice that performs this function, such as "Full Size" or "Resize".)
- 4. Set the QuickSim II session working directory to be *qsim\_n* by performing these steps:
  - a. Choose the following QuickSim II pulldown menu item:

### (Menu Bar) > MGC > Location Map > Set Working Directory

The SET WOrking Directory prompt bar appears.

b. Enter the following path in the prompt bar Directory field:



c. Click on the **OK** button.

The new working directory is set. Any future commands requiring a pathname will assume this is the current directory. Subsequent steps will show how you can use either leafnames based on the working directory or fully qualified pathnames.

5. Create the schematic view window as follows:



The schematic view window appears. The circuit should be centered in the window. If it isn't, center it using the scroll arrows or the View All function key. Refer to the softkey area for the location of the View All function key.



- 6. Run the "sim.do" batch simulation file for this circuit as follows:
  - a. With your pointer anywhere in the QuickSim II session window, type the following command:

Schematic\_view dofile sim.do



Because you set the working directory in an earlier step, you only need to provide the relative path of the sim.do file. However, if the "working directory" is not defined within the session, you must provide the complete pathname such as, \$HOME/training/qsim\_n/LATCH/sim.do

b. Press the Return key to execute the command.

The simulation runs, until time 4650 when the dofile finishes. Also, the schematic view is adjusted. A List window was created at the beginning of the simulation run, which now contains the results. The following figure shows some of this data.

	Li	st				- [	
4350.0	1	1	1	0	Х	Х	$\underline{\Delta}$
4400.0	1	1	0	0	0	1	
4450.0	X	1	Х	0	0	1	
4500.0	X	1	0	0	Х	X	
4550.0	1	1	1	1	1	0	
4600.0	1	1	0	0	1	0	
4650.0	1	0	1	0	1	0	
Time(ns)	^/C	LR ^/F	^/( PRE	CLK ^/[	( ^/( )	ک /^	
							· ′

- 7. Open a Trace window as follows:
  - a. Click on each of the signal names at the bottom of the List window to select them.



b. Click on **TRACE** in the common palette area.

The Trace window appears. The default location for this window is the bottom of the QuickSim II session window. The selected signals are placed in the Trace window.

- 8. Open the session Transcript window as follows:
  - a. Click on: **SETUP** > Show Transcript

The Transcript window overlays the other windows. Examine the contents of this window. Note that functions have been transcripted for the menu items you have entered. These functions begin with "\$" and "\$\$".

- b. Adjust the size of the Transcript window using the window grow borders so that only the Trace window and Transcript window are visible.
- c. Click on the Minimize button in the Transcript window to make it into an icon. What is the name of the icon that is created?

Note that the palette has disappeared because the Transcript icon is active. Not all QuickSim II windows use palettes.

d. Activate the List window by clicking the stroke (middle) mouse button in it. The palette is again visible.

- 9. Add a Monitor window, as follows:
  - a. Make sure that the six signals shown in the List and Trace window are still selected.
  - b. Click on: DBG GATES
  - c. Adjust the size of the monitor so that the Transcript window icon is visible. Use the window sizing handle at the right end of the Monitor window.

What is the time displayed in the Monitor Window?\_\_\_\_\_

10. Delete the Transcript window icon by positioning the pointer on the Transcript icon, accessing its popup menu, and choosing:

#### (Transcript icon) > Close

11. Examine the current version of the Falcon Framework as follows:

#### a. Choose: (Menu Bar) > Help > More Help > On Support Center

The Customer Assistance Information box appears with information on how to get Mentor Graphics assistance. This information also includes the release information for the Falcon Framework.

b. Close the Customer Assistance Information window.

- 12. Exit QuickSim II as follows:
  - a. Choose: (QuickSim II Window Menu ) > <u>C</u>lose (In some window environments, this may be the "Quit" menu item).

The Exit QuickSim II dialog box appears.

- b. Click on the "Save Setup" button to unhighlight it. You will not save setup condition in this lab exercise.
- c. Verify that the "After Saving" button and the "Save 'results' Waveform DB" button are selected. The dialog box should be setup as follows:

uickSim				
Without saving				
Save Setup (WDBs, Windows, etc)				
Save 'results Waveform DB				
set Cancel				

d. **OK** the dialog box.

The Save Waveform DB dialog box appears asking you for a path at which to save the "results" waveform database.

Name: results 1. Click
Viewpoint 2. Fill in
Leafname results_lab1
Replace

e. Fill out the Save Waveform DB dialog box as follows:

f. A Question box appears, asking you if you want the viewpoint to be persistent (create the viewpoint object on disk). This is because the viewpoint object that QuickSim II created on invocation only exists in memory, not on disk.

	Question
	The current viewpoint must be persistent to continue the 'Save WDB command.
?	Do you wish to save the current viewpoint?
Click	Yes No

g. Click on the **YES** button to save the viewpoint to disk.

The message window indicates that "results" is saved to the viewpoint. QuickSim II exits, returning you to the Design Manager. You can also remove the Terminal transcript pad.

- 13. Examine the *qsim\_n* database for new objects as follows:
  - a. Use the Design Manager iconic navigator window to view:

\$HOME/training/qsim\_n/LATCH

Notice the new design viewpoint object named default. It was created when you made the Viewpoint persistent (saved it as a design object).

b. Navigate beneath the "default" design viewpoint object. Examine its contents.

Note the shape of the icon used for the "results\_lab1" waveform database. This icon indicates that this object is a waveform database.



14. Exit the Design Manager as follows:

Choose: (Session Window Menu) > <u>Close</u> (In some window environments, this may be the "Quit" menu item).



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 1 Solutions" on page A-1.

Then continue on to the next module.

## **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 1 Solutions" on page A-1.

- 1. Name the common simulation user interface: \_\_\_\_\_\_
- 2. True or False. It is mandatory to have a design viewpoint to use QuickSim II on the design?
- 3. What two applications automatically create required design objects when QuickSim II is invoked?
  - 1.\_\_\_\_\_
- 4. Name the two parts of an electronic design.
  - 1.\_\_\_\_\_
  - 2.\_\_\_\_\_
- 5. What Run options are available with the [palette] Run button?
- 6. To stop the simulation on your workstation during a run, use the \_\_\_\_\_ key.
- 7. When you exit QuickSim II, what information are you prompted to save:

# **Module Summary**

Module 1 introduced the concepts and features that are in the Mentor Graphics Electronic Design Automation environment. These are grouped into the following categories:

• **Design Manager and Electronic Designs**. Designs are made up of two parts: a component hierarchy which contains the models, and a Viewpoint which contains the configuration rules for the design. The Electronic Design Database provides a flexible structure for making changes to design.

The Design Manager handles the copying, releasing, version managing and checking of designs. The QuickSim II application can be invoked from within the Design Manager. An Integrated Design Manager is also available from within most applications.

- Application architecture. The Falcon Framework provides a common user interface and support structure for all of the Mentor Graphics applications. The Common Simulation User Interface (SimView/UI) provides a common look and feel to all of the analysis applications.
- User interface. QuickSim II menus, palettes, function keys, windows, and commands provide many of the desired features of a digital simulator. For most operations you want to perform, there are several way to perform them using the different interface options. Use the method that you find most productive.
- **Runtime Kernel.** The kernel performs the analysis part of the simulation. The Run command allows some flexibility, including running for time, until time, and until stop. There is a default kernel configuration mode that disables timing and checking, to allow fast functional analysis. More on kernel configuration is in Module 4.

In the next module, Module 2, you will learn the concepts necessary to create and use waveform databases.

# Module 2 Waveform Database Concepts

Overview	2-2
Lesson	2-3
QuickSim II Architecture	2-4
QuickSim II Design Flow	2-6
Cross-Window Selection	2-8
Waveform Databases	2-10
Predefined Waveform Databases	2-12
Default Waveform Databases	2-14
Waveform Database Architecture	2-16
Waveforms	2-18
Basic 12-State Logic Events	2-20
Other Logic Event Types	2-22
Waveform Storage Formats	2-24
Loading Waveform Databases	2-26
Saving Waveform Databases	2-28
Unloading Waveform Databases	2-30
Lab Preview	2-32
Lab Exercise	2-34
Procedure 1: Loading Waveform Databases	2-34
Procedure 2: Waveforms vs. Pin and Net Activity	2-43
Procedure 3: Saving Waveform Databases	2-45
Test Your Knowledge	2-47
Module Summary	2-48

### **Overview**



### Lesson

On completion of this module, you will be able to:

- Describe the QuickSim II application architecture, and the individual elements that are needed to perform a digital simulation
- Describe how waveform databases are stored as results, loaded into memory, and connected to a design for stimulus.
- Name the states that QuickSim II digital simulation supports, in both 12- state mode, 4-state mode, and other modes.
- Be able to load and connect waveform database as stimulus in a simulation.
- Describe the waveform database default designations.



You should allow approximately 1 hour and 10 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.



## **QuickSim II Architecture**

- **DVAS Design Viewing and Analysis Support**
- **DVE Design Viewpoint Editor**
- **DSC Design Syntax Checker**

# **QuickSim II Architecture**

The figure shows a representation of the simulation architecture. The following list defines the major elements of the simulator architecture:

- **Front end**. The front end provides the communication link between the user (through the session windows, keys, and mouse), the design data, and the QuickSim II kernel. The front end uses the following features to develop its look and feel.
  - **QuickSim II Userware**. Consists of menus, key definitions, session windows, and gadgets, that enhance the usability of the simulator.
  - **SimView/UI (Common Simulation User Interface)**. A set of commands (common to all Mentor Graphics simulators) that let you interact with the simulation and display the waveform database information for analysis.
  - **DVAS (Design Viewing and Analysis Support).** Lets you select, view, group, and report on design items.
  - **DVE (Design Viewpoint Editor)**. Lets you perform incremental design changes during the QuickSim II session.
  - **DSC** (**Design Syntax Checker**). Lets you check simulation properties during the simulation application session. The DSC is useful after making incremental design changes.
- **QuickSim II Kernel.** Performs QuickSim II-specific operations. This element of the architecture performs the actual simulation by analyzing the functional and timing models and by responding to input stimulus.
- Electronic Design Database. Contains the design structure, design viewpoint, back annotations, and other design objects that QuickSim II uses. This database provides information to both the front end and the kernel.
- **Waveform Databases.** Provides the stimulus to the design and holds the simulation results. The kernel generates the results waveform database, and the front end accesses this waveform database for display purposes.



- 1. Prepare unique models and access Mentor Graphics component libraries
- 2. Capture design in Design Architect
- 3. QuickSim II automatically creates design viewpoint or use DVE manually
- 4. TimeBase automatically calculates delay values or can be invoked as an optional step
- 5. Invoke the simulator, setup timing trade-offs, and run the simulation
- 6. Modify property values as back annotations
# **QuickSim II Design Flow**

The design flow for a typical simulation is as follows:

- 1. Prepare any unique subcomponents to be used in the design. This can easily be done by modifying existing Mentor Graphics library components. Other unique component models may be ASICs or functional blocks.
- 2. Capture the design using Design Architect, which provides schematic and symbol editing capabilities. You can use System-1076 modeling to create textual descriptions of the design, and then use synthesis tools to create discrete logic from these descriptions.
- 3. Optionally, create a design viewpoint with a custom configuration using the Design Viewpoint Editor (DVE). A design viewpoint is a versioned object that contains the rules that configure your design. Running DVE is optional because the simulator (upon invocation) creates a design viewpoint with a default configuration if one does not already exist.
- 4. When you invoke QuickSim II, TimeBase automatically calculates actual delay values from the information and equations provided in technology files. You can optionally use TimeBase to pre-build timing when complex timing equations are included in the design. The resulting timing data cache is placed in the design viewpoint. Then timing is not rebuilt each time you invoke QuickSim II.
- 5. Invoke the simulator, set up the simulation, and run the simulation. When you invoke the simulator, you can specify many setup conditions such as the timing, spike handling model, and several runtime checking capabilities. Simulate the design with various combinations of timing modes (such as unit delay, min, typ, and max) and timing checks.
- 6. Optionally, you can modify design property values to correct design errors or to perform "what if" analysis. For example, you can change Temperature, Voltage, or Cap\_net property values to immediately change the results of timing equations that use those properties. Note that the simulator places all property changes that you make interactively in a back annotation object.

# **Cross-Window Selection**

- Cross-highlighting
  - By name (text) or graphic selection
  - By time. View conditionals
- Source viewing. Creates window or adjusts view

Types. Schematic, Report, (VHDL with SimView/UI and simulator)



To remove same name in a window--select gadget instead of name, and then delete.

## **Cross-Window Selection**

Representations of the same object are logically associated. For example, a net, the pin to which it is attached, its entries in the List and Trace windows, and its name in error messages are all related. Thus, if you select one representation of an object, all representations of that object are selected and highlighted. This is called *cross-highlighting*.

For example, if you select a pin name in an error message, the pin is highlighted on the design view, and the pin name is highlighted in the List and the Trace windows.



If you have added a pin or net name to a window several times, and want to delete only one of the additions, select the gadget handle and then execute the delete operation.

You can also use cross-highlighting when you select by time. For example, if you get a spike error message and select the time given in the message, you can issue the View Time command in the List or Trace window to view the conditions of various pins or nets at that time. The same time label in the other windows will also be highlighted.

Use cross-highlighting when you get an error message to go directly to the Trace, List, or design representation of the trouble spot, and make changes before simulating again.



For more information on cross-highlighting, refer to "Using Cross Highlighting" in the *SimView Common Simulation User's Manual*.

# **Waveform Databases**



- Can be viewed and edited
  - To view -- load into memory; add to window
  - To edit -- load into forces waveform database

#### **Waveform Databases**

Waveform databases are a compact form for storing simulation stimulus and results. They are designed to contain, manage, and save one or more waveforms. QuickSim II interacts only with waveform databases, translating all other forms of stimulus, such as force files (Force commands) and logfiles to the waveform database format before they are used. Waveform databases have the following characteristics:

- They are a binary form of the values that are associated with a pin or net. This is a particularly fast form of stimulus, because the data is event-ordered.
- They can be merged. If you create several waveform databases to use for stimulus and want to merge them into one waveform database, you load them into memory and connect them to the stimulus waveform database. As you connect them, you can specify time offsets to shift the point in time where the waveforms are applied.
- They can be viewed and edited. To view the contents of a waveform database, load it into memory and then add the desired waveforms to the Trace, List, or Monitor windows. To edit a waveform database, first load it either into the "forces" waveform database or as a user-defined waveform database, which ever is to be designated as the "Force Target." Then, issue Force and Delete Force commands to change the desired waveforms. The graphical waveform editor can also be used to modify waveforms in a Trace window.
- Any waveform database can be the "default." When QuickSim II displays a waveform, evaluates expressions, or evaluates actionpoints, it uses the waveforms in the default waveform database, unless the waveform name is prefixed with the name of the waveform database in which it resides. Page 2-14 describes the default waveform database.
- They can be saved to disk. EXCEPTIONS: the stimulus waveform database can't be written to disk in waveform database format (only as forcefile format) and the asserts waveform database can only be saved by saving the setup.
- They are the source for logfiles and force files. You can translate waveform databases loaded into program memory, including the "results", "stimulus", or "forces" waveform database.

# **Predefined Waveform Databases**



waveform database

- Default Force Target for commands and functions
- Waveforms automatically connected to design



waveform database

- Merges stimulus to kernel with time offsets
- Cannot save as waveform database to disk



waveform database

- Holds all kept simulation results
- Used to display, in expression or breakpoint



waveform database

- Created when first Assert command is issued
- Holds assertion waveforms (expected results)
- Used to analyze actual results
- Can be saved as part of setup data

## **Predefined Waveform Databases**

The following special purpose waveform databases are automatically created and managed by QuickSim II:

- 'forces' waveform database. This unique database contains waveform data that can be created or modified by the Force command. You can load any waveform database from the disk into the forces waveform database. By default, the forces database is assigned as the "Force Target" and is connected to the design, although it can be disconnected.
- 'stimulus' waveform database. This unique database merges and supplies to the simulator all the stimulus being applied (connected) to the design.

The kernel deals exclusively with the stimulus waveform database when reading stimulus, although you can connect any number of stimulus-providing waveform databases. The stimulus waveform database acts like a funnel, merging and managing the waveforms from all connected waveform databases, presenting a single stream of events to the design. You can connect any waveform database that is loaded into memory (except the results and asserts databases).

Through the merging capabilities of the stimulus waveform database you can also use offsets to shift a given waveform either forward or backward in time. You can only save the stimulus waveform database to disk as a forcefile.

• **'results' waveform database**. This unique database always holds the simulation results that the simulator keeps for display, expression, or breakpoint evaluation.

This waveform database is the "Default" when you invoke QuickSim II. The simulator looks in the results waveform database for data when it adds a signal to the Trace, List, or Monitor window, or when it evaluates an expression or breakpoint. You can save the results waveform database with the design.

• 'asserts' waveform database. This unique database is created when an assertion test is created. The database holds the assertions (sometimes called expected results). During a simulation run these assertions are compared against the actual results contained in the results database. The comparison results can be viewed in the Trace or List window to see the differences found.

# **Default Waveform Databases**

- "Default" waveform database
  - Used if none is specified
  - SimView default is the forces waveform database
  - QuickSim II (under SimView/UI) default is the results waveform database
  - You can define any waveform database as the default
- "Force Target" waveform database
  - Used by Force commands and waveform editor if no waveform database is specified
  - Default Force Target is *forces* waveform database
  - You can define any waveform database except stimulus and results as the Force Target

## **Default Waveform Databases**

Two default settings are available for the waveform databases currently loaded into memory: "default" and "force target."

- "default" waveform database. The waveform database that is assigned as the default is used by any subsequent implicit waveform database actions that are not Force or waveform editor related. An implicit waveform database action is one that does not explicitly specify a waveform database by name. Upon invocation, SimView sets the "forces" waveform database as the default; for other SimView/UI-based applications, the "results" waveform database is the default. You can set any waveform database that is currently loaded into memory as the default.
- "force target" waveform database. The waveform database that is assigned as the force target is used by any subsequent implicit waveform database actions that are Force command or waveform editor related. To edit a waveform, it must first reside in the database assigned as the Force Target or you must specify the complete waveform name (including the waveform database name). Upon invocation, SimView sets the "forces" waveform database as the force target. You can set any waveform database that is currently loaded into memory (except the "stimulus" and "results" waveform databases) as the force target.

You can simplify subsequent waveform database operations by setting the "default" and "force target" waveform databases and by not specifying an explicit waveform database. Subsequently-displayed waveform database dialog boxes show the default or force target waveform database currently assigned.

### **Waveform Database Architecture**



- Waveform Databases contain waveforms
- Waveforms contain time-ordered events
- Events either are retained in waveforms, associated with a pin or net in a design, or both

# Waveform and Waveform Database Architecture

Waveform databases are design object containers that can be saved to disk or loaded into program memory from disk. Waveform databases contain one or more waveforms.

Waveforms contain events that can be applied to a design circuit as stimulus or that can be extracted from a design circuit as results. Stimulus waveforms are those that are applied to a design. Results waveforms are those that are extracted from a design.

Events are ordered pairs of values that represent electrical activity in a design circuit. For digital simulators each pair of values consist of a state value (1, 0, or X; high, low, or unknown) and of a time value at which the state is to become active. The events can be applied through a connected stimulus waveform and can be kept as a results waveform. If a waveform is not connected or if the events are not kept, the events are processed by the simulator and then discarded; no waveform is available for viewing. You must explicitly, or implicitly, connect or keep a design's event activities to be able to view them.

# Waveforms



- An Event = Ordered Pair of Time/State Values
- A Waveform = Set of Ordered Events
- Can be Traced and Edited
- Waveforms are stored in a Waveform Database
- Waveform Database waveforms are in Binary (fast form of stimulus)

### Waveforms

Waveforms are designed to contain and manage event information and to reside in waveform databases. QuickSim II interacts only with waveform database waveforms, translating all other forms of stimulus, such as Force commands, forcefiles, and logfiles, to the waveform database format before they are used.

Waveform Database waveforms have the following characteristics:

- They are a binary form of the values that are associated with a pin or net. This is a particularly fast form of stimulus, because the data is event-ordered.
- They can be merged. If you create several waveforms to use for stimulus and want to merge them into one waveform database, you load them into memory and connect them to the stimulus waveform database. Once connected, you can specify time offsets to shift the point in time where the waveforms are applied.
- They can be viewed and edited. To view the contents of a waveform, load it into memory and then add the desired waveforms to the Trace, List, or Monitor windows. To edit a waveform, first load it into a waveform database. Then, issue Force and Delete Force commands to change the desired waveforms. The graphical waveform editor can also be used to modify waveforms.

To effectively edit waveforms using QuickSim II you should have a good understanding of the waveform database default settings: Default and Force Target. Both of these are discussed later in this module under "Waveform Storage Formats" on page 2-24.

# **Basic 12-State Logic Events**

#### Mentor Graphics 12-state simulators:

	Signal Level			
Drive Strength	Low (0)	High (1)	Unknown (X)	
Strong (S)	0S	1S	XS	
Resistive (R)	0R	1R	XR	
High Impedance (Z)	0Z	1Z	XZ	
Indeterminate (I)	01	11	XI	

- TTL designs -- require only five state values
  - 0 and 1 (for driving devices)
  - X (for either 1 or 0, but you don't know which)
  - 1R (for pullup resistors)
  - XZ (for any high-impedance signal level)
- ECL pulldown resistors -- substitute 0R for 1R
- MOS designs -- indeterminate signal strength
- Fixed drive -- different from the simple strong (S)
  - Cannot be overridden by contending signals
  - Used for voltage sources (VCC, VDD, VSS) or ground (GND)

# **Basic 12-State Logic Events**

Mentor Graphics digital logic simulators use three logic values: 0, 1, and X. The X value represents a logic value that could be *either* 0 or 1, but cannot be reliably determined. X logic values can occur at design "power-up" or as a result of signal contention, where competing logic values are driven simultaneously onto the same net.

Signal drive strengths allow a simulator to accurately resolve signal contention and to simulate the subtle effects of different design technologies. The simulator uses four signal drive strengths: strong (S), resistive (R), high impedance (Z), and indeterminate (I). It combines the signal drive strengths with the three logic values to create the twelve signal states required for comprehensive and accurate simulation of the different design technologies.

TTL designs typically require only five of these logic value/signal strength combinations: 0 and 1 (for driving devices), X (for either 1 or 0, but you don't know which), 1R (for pullup resistors) and XZ (for any high-impedance signal level). By substituting a 0R in place of the 1R, you can accurately model ECL and its pulldown resistors. The need to accurately model MOS designs at the transistor level led to the *indeterminate* signal strength, enlarging the state-strength table to 12 combinations. The table on the opposite page shows the resulting combination map used by QuickSim II.

QuickSim II uses an additional drive strength that cannot be overridden by contending signals, which allows you to simulate a driving positive voltage level (VCC) or ground level (GND).

This overriding drive condition is a *fixed* drive, and is different from the simple strong (S) drive. For example, when the 1S and 0S signal states are combined, the result is XS, or the unknown signal state. However, a fixed signal state of 1S, which you would use to model a VCC connection, always overrides any other contending signal state during a simulation (stays "fixed" at 1S in this example). A fixed signal state is also useful in debugging because it cancels the effects of any driving output that is connected to the net.

# **Other Logic Event Types**

- Mentor Graphics 12-state events (double or single character representation)
- VHDL 9-state events

#### • Qsim 4-state events

12-state			
Double Character	Single Character	9-state	4-state
X (Xs)	Х	'U', 'X', '-'	Х
0 (0s)	0	'0'	0
1 (1s)	1	'1'	1
Xz	Z	'Z'	Z
Xr	W	'W'	
Or	L	'L'	
1r	Н	'H'	
0z	Ν		,
1z	Ρ		
Oi	J		
1i	К		
Xi	М		
of (off)	Q		

# **Other Logic Event Types**

Mentor Graphics digital logic simulators use three types of events within waveforms: 12-state, 9-state, or 4-state.

12-state represents the full set of state values that can be processed by QuickSim II or any of the other Mentor Graphics digital logic simulators.

9-state (or mvl\_9) is used by VHDL and is mapped to the QuickSim 12-state values for simulation purposes.

4-state (or qsim\_state) is used by System-1076 and is also mapped to QuickSim 12-state values.

Single character mode can be used in QuickSim II to provide a direct mapping from 9-state and 4-state to 12-state.

# **Waveform Storage Formats**



- All waveform formats are converted to Waveform Database form when loaded into program memory
- Formats recognized
  - Waveform database binary format
    - Binary form of signal logging
    - Useful for partitioned designs
    - Connect multiple waveform databases as stimulus
    - Interactively edit during session
  - Logfile format
    - An ASCII file
  - Forcefile format
    - Dofile containing Force commands

### **Waveform Storage Formats**

The simulator converts all forms of stimulus into waveform database format before actually scheduling the stimulus events. The simulator then deals directly with the waveform database, which is more efficient than interpreting Force commands.

You can *force*, or apply stimulus, to any net in the design. When you do, the simulator schedules a force event using the logic state (1, 0, X), signal strength (S, R, Z, I), and time you provide.

- Waveform database. Using a waveform database for stimulus is similar to using a logfile, because it works best with partitioned designs, yet waveform databases are significantly faster during execution.
- **Logfile**. An ASCII file. Logfiles can be used by third party tool users as a method of exchanging simulation data. Before using a logfile, you must load it using the Load Log command.
- **Forcefile**. A dofile that contains Force commands. You can submit forcefiles as a batch of stimulus. Forcefiles are useful once you have determined and verified a set of Force commands. To create a forcefile, you can paste Force commands from a transcript, or enter them manually.

Force commands are the most common form of stimulus. Force commands are automatically put into waveform database format in the Force Target waveform database. You can also issue forces by choosing menu items, palette icons, or manually typing them in the popup command line.



Forces are discussed in Module 7 "Creating and Modifying Stimulus".

# **Loading Waveform Databases**

#### Load three kinds of waveform data format



• or enter: DOFile forcefile\_name

# **Loading Waveform Databases**

Waveform data must be created in or loaded into program memory to be used by QuickSim II. When you load waveform data, it is copied from disk, translated to waveform database format if necessary, and placed into program memory.

Waveform database or logfile formats are loaded into the QuickSim II program memory by choosing the **File > Load > Waveform DB** pulldown menu item. The menu item displays the Load Waveform DB dialog box that assists you in completing the load operation. Some key choices in the dialog box include:

- **Pathname** An entry box and Navigator to identify the waveform data object on disk that you want to load.
- **Load into 'forces'** A button that loads the waveform data from disk into the forces waveform database in program memory. Normally this selection is used if you plan to edit a waveform.
- **WDB name** An entry box that allows you to custom name the waveform database you want created in program memory.
- **Connect Waveform DB immediately** A button that automatically connects as stimulus the waveforms contained in the waveform database.
- Forcefiles cannot be *loaded* using the direct load method; forcefiles contain Force commands that must be executed. Therefore, to load a forcefile you must either choose the **Setup** > **Force** > **From File** pulldown menu item or you must use the Dofile command as you would for any other executable dofile.



• To save the stimulus waveform database, enter the SAVe LOg - or - SAVe FOrcefile command

# **Saving Waveform Databases**

Once you or the simulator create a waveform database, you can make it *persistent*; that is, you can make it permanently available for future simulation sessions by saving it to disk. You are normally asked if you want to save these objects when you exit QuickSim II.

You can save all waveform databases in waveform database format except the stimulus and asserts databases. Both of these databases are specially created by the simulator and currently cannot be saved in waveform database format.

The stimulus waveform database can only be saved in logfile or forcefile formats. To do so you must use either the Save Logfile or Save Forcefile command.

The asserts waveform database can only be saved as part of the QuickSim II setup. To save the asserts waveform database as part of the current setup, choose the **File > Save > Setup** pulldown menu item or issue the Save Setup command. In either case, once an asserts waveform database is saved as a setup object, you can restore asserts by using the **File > Restore > Setup** pulldown menu item

Saving a waveform database to disk does not unload it from QuickSim II program memory. To remove a waveform database from program memory, use the Unload feature discussed next.



For additional information about how to save waveform databases refer to "Manipulating Stimulus" in the *SimView Common Simulation User's Manual.* 

# **Unloading Waveform Databases**

#### Removes waveform databases from program memory



## **Unloading Waveform Databases**

Unloading a waveform database deletes the database or just the database contents from program memory depending on which waveform database you're unloading.

All user-defined and asserts waveform databases are deleted in their entirety from program memory. The predefined waveform databases, forces, stimulus, and results, are each handled differently.

The forces waveform database only allows you to unload the waveforms contained within the database; the waveform database itself remains in program memory awaiting new waveforms.

The stimulus waveform database cannot be unloaded. A similar result can be achieved by choosing the **Setup > Disconnect > Waveform DBs** pulldown menu item. When the waveforms connected to the design are disconnected, they no longer are included in the stimulus waveform database.

The results waveform database cannot be unloaded either. Because the results waveform database contains the kept data from time 0 to the current simulation time, you can delete the contents by using the Reset State command or you can delete the waveforms in the database by deleting the *keeps*.

The Unload Wdb command contains an option that allows you to save the waveform database to disk before deleting it from program memory.

# Lab Preview

- Invoke QuickSim II on the LATCH design using the Design Manager
- View the contents of a forcefile
- Load various formats of waveform data
- Graphically view the loaded waveform events
- Graphically view the waveform events on a design net (as the result of a run)
- Save the stimulus and results waveform data
- Exit QuickSim II

## Lab Preview

In the lab exercise for this module, you will:

- 1. Invoke QuickSim II on the LATCH design using the Design Manager.
- 2. View the contents of a forcefile by using the Notepad editor. You will also see how the various formats are represented in the Notepad Navigator.
- 3. Load various formats of waveform data into QuickSim II program memory.
- 4. Graphically view the loaded waveform events in a Trace window.
- 5. Graphically view the waveform events on a design net. These will be traces due to a simulation run.
- 6. Save the stimulus and results waveform databases to disk to make the waveform data persistent.
- 7. Exit QuickSim II without saving any further results.

### Lab Exercise



If you are reading this workbook online, you might want to print the lab exercises to have them handy when you are at your workstation.

#### **Procedure 1: Loading Waveform Databases**

1. If necessary, log into your workstation and set your current directory to your *qsim\_n* directory by entering the following.

shell> cd \$HOME/training/qsim\_n

Note: If you did not create the training directory in \$HOME, be sure to use your unique path instead of \$HOME.

- 2. Invoke QuickSim II on the LATCH design from within the Design Manager by performing the following:
  - a. Invoke the Design Manager by entering:

shell> \$MGC\_HOME/bin/dmgr

- b. Using the Select mouse button, click on the icon highlights.
- c. Choose the following Navigator window popup menu item:

#### (Navigator popup) > Open > QuickSimII

3. Maximize the QuickSim II session window using the Maximize button, (or the "Full size" window menu option).

- 4. Set the QuickSim II session working directory to be *qsim\_n* as follows:
  - a. Choose the following QuickSim II pulldown menu item:

#### (Menu Bar) > MGC > Location Map > Set Working Directory

b. Enter the following path in the prompt bar Directory field:

SET WO D	Directory	\$HOME/training/gsim_n	OK	Cancel
0	2	•••••=···=··		Canoci

c. Click on the **OK** button.

The new working directory is set.

5. Report the current waveform databases in program memory by choosing the (Menu Bar) > Report > Waveform DBs pulldown menu item.

The Waveform DBs report window is displayed as shown here.

□ Waveform DBs 1			
WDB name	Default	Force Target	$\Box$
forces	F	Т	
results	Т	F	
stimulus	F	F	

What are the session default waveform databases?

- Default Waveform Database: \_\_\_\_\_\_

- 6. Report the waveforms contained in each of the predefined waveform databases by performing the following steps:
  - a. Select each of the waveform databases listed in the Waveform DBs report window.
  - b. Choose the (Waveform DBs popup) Report Waveforms menu item.

The forces, results, and stimulus waveform report windows are displayed (one on top of the other). Each of the report windows are empty because you have not loaded or created any waveforms in this session.

- 7. Relocate the forces, results, and stimulus waveform report windows so that each is visible within the session. You will use these report windows later in this procedure.
- 8. Now let's look at the waveform data that is available on the disk by using the Notepad "Select file to view" navigator as follows:
  - a. Choose the (Menu Bar) > MGC > Notepad > Open > Read-only... pulldown menu item.

The "Select file to view" dialog box appears.

b. Click on the Navigator button.

The File Navigator dialog box is displayed.

- c. Click on the filter button to disable filtering.
- d. In the Filter Objects dialog box, select all of the listed entries under "Types To Display" and click on the "Remove Selected Type" button.
- e. OK the dialog box.
- f. List the objects in LATCH by first selecting **C** LATCH, then

selecting the **J** button.

The Notepad navigator dialog box is redrawn to display the contents of the LATCH design. This shows only the information on disk, not what is in program memory.

g. Using the icons in the Notepad navigator dialog box as identifiers, draw lines that connect the file names to the name of the waveform format (type of object) that each icon represents (page 2-24):



h. View the contents of the forcefile, *forces\_forc*, in a notepad by selecting the filename and clicking the **OK** button. Also **OK** the "Select file to view" dialog box.

A Notepad window similar to the one shown here is displayed:

Notepad -\$HOME/training/qsim_n/LATCH/forces_forc (R)	1
// SET USer Scale -type Time 1e-09	$\square$
// SETup FOrce -Charge	
FORCe /CLR 1 0.0 -Charge -Abs	
FORCe /PRE 1 0.0 -Charge -Abs	
FORCe /PRE 0 325.0 -Charge -Abs	
FORCe /PRE 1 375.0 -Charge -Abs	
FORCe /CLR 0 425.0 -Charge -Abs	
FORCe /CLR 1 475.0 -Charge -Abs	
FORCe /PRE 0 660.0 -Abs	
FORCE /PRE 1 760.0 -Abs	

As you can see, a forcefile is simply an AMPLE file that contains Force commands. Notice that "minimum typing" is shown by the capitalization in the FORCe command. For complete information about the Force command and its syntax, refer to the *SimView Common Simulation Reference Manual*.

- 9. Resize the Notepad window so that it and the three waveform report windows are visible.
- 10. Load the waveforms represented in the *LATCH/forces\_forc* forcefile by performing the following steps:

a. Activate a window other than the Notepad.

#### b. Choose: (Menu bar) Setup > Force > From File

The Load Forcefile dialog box is displayed.

- c. Use the **Navigator...** to select the **forces forc** forcefile.
- d. **OK** the Navigator and Load Forcefile dialog boxes.

As the forcefile is executed the forces waveform database is loaded with waveform data; and because all forces are automatically connected as stimulus, the stimulus waveform database is also loaded.

- 11. Create a Trace window containing the forces waveform database waveforms by performing the following:
  - a. Select the waveforms in the forces waveform report window.
  - b. Use the  $\neg$  stroke to add the waveforms to the Trace window.

A Trace window is displayed containing two waveforms. What are the names of the two waveforms shown in the Trace window?

- 1. \_\_\_\_\_
- 2.\_\_\_\_\_

As you can see, the names of the waveforms reflect that they are located in the forces waveform database.

12. Compare the Force commands in the *forces\_forc* forcefile with the waveforms graphically shown in the Trace window. Notice that each transition is represented. Shown here are the transition comparisons for CLR:



- 13. Load the remaining input waveforms into a second, user-defined, waveform database by performing the following steps:
  - a. Click on the following palette icon:



The Load Waveform DB dialog box is displayed as shown here:

Load Waveform DB				
☐ Viewpoint				
Pathname Navigator				
Load into the 'forces' WDB				
WDB name				
<b>Repeat</b> (Used with Logfiles only)				
Should the WDB be connected now for use as stimulus? If not, the WDB may be connected later using the "Connect WDB" command				
OK Reset Cancel Help				

- b. Use the **Navigator...** to select the **forces** waveform database file.
- c. **OK** the Navigator dialog box; the path to the forces file is displayed in the Pathname entry field.
- d. Click on the **Load into the 'forces' WDB** button.
- e. Click on the **OK** button.

The session reports the following message because you cannot load waveform data into a waveform database that already contains data:

Waveform database 'forces' is not empty. Unload WDB before overloading.

You must either unload the data in the forces waveform database or load the data into a different waveform database. We'll use the latter.

- f. Repeat steps a through c to again display the Load Waveform DB dialog box and select the forces file.
- g. This time load the waveform data into a unique user-defined waveform database called mywdb by entering the following:

Load into the 'forces' WDB < 1. Do not Select	
WDB name mywdb	
Repeat (Used with Logfiles only) 2. Enter name	
Should the WDB be connected now for use as stimulus? If not, the WDB may be connected later using the "Connect WDB" command	
Connect Waveform DB immediately	oses er of
Connect Parameters the dialog bo	(X)
Offset Start Stop 1000 4. En	ter clock
Absolute times <a>F</a> Merge with existing waveforms	
Force type for the connection = Default Select	
5. Click on	
OK Reset Cancel Help	

The "mywdb" waveform report window appears. The report window lists the waveforms just loaded into memory; and because you selected to connect immediately, the stimulus waveform database is also updated.

- 14. Add the new waveforms to the Trace window by performing the following:
  - a. Select the waveforms in the mywdb waveform report window.
  - b. Use the trace stroke  $\blacktriangleleft$  to add the waveforms to the Trace window.

You have completed the two methods of loading waveform data into session waveform databases and you have connected the data to the design as stimulus. Although we did not load a logfile format, to do so you would use the same procedure as was used for the *forces* file.
#### **Procedure 2: Waveforms vs. Pin and Net Activity**

In the previous lab procedure you traced waveforms that you loaded into the forces and mywdb waveform databases. In this procedure you will trace the input and output nets of the design on which you invoked.

15. Create the schematic view window by clicking on the following palette icon:



The schematic view window containing the LATCH design is displayed.

16. Select all of the nets in the LATCH schematic view window by clicking the Select mouse button on each net port \_\_\_\_\_ in the design.



As shown in the figure, the nets should turn to white dashed lines when they are selected. If you make a mistake and select the wrong items, click the Select mouse button on the object you selected by mistake; object selection toggles.

17. Add the selected nets to the Trace window by using the  $\neg$  stroke.

								Tr	rac	e													
forces@@/CLR +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	$\square$
forces@@/PRE +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
forces@@/CLK +	÷	ł	ł	ł	-	+	ł	t	t	ł	÷	-	÷	ł	÷	ł	t	ł	÷	+	÷	ł	
forces@@/D+	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
/PRE +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
<mark>/D</mark> +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
/CLK +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
/CLR +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
<mark>/Q</mark> +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
<mark>/QB</mark> +	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
0		100		200		300		400		500 Ti	me	600 e(ns	)	700		800		900	1	1000	1	100	$\nabla$
$\triangleleft \square$																						$\exists \succ$	

The Trace window should now look similar to the one shown here:

Unlike the waveforms in the forces and mywdb waveform databases, the nets do not show any waveform activity. This is because you have not run the simulator yet, so there is no result data. The "results" report window now contains waveforms for each of the nets traced.

18. To see a "history of events," run the simulator 500 nsec as follows:



You can see that the Trace window is updated to show the waveform data as it occurs on the various input nets. You can also see the resulting waveform data as it occurs on the output nets: /Q and /QB.

What are the current values of /Q \_\_\_\_\_ and /QB \_\_\_\_?

19. Complete the simulation of the currently connected stimulus by running the simulator another 500 nsec. To do so enter the same command as in the previous step.

#### **Procedure 3: Saving Waveform Databases**

This procedure describes how to save waveform databases.

- 20. Save the results waveform database to the design viewpoint by performing the following steps:
  - a. Save "results" in the waveform database format by clicking on the following palette icon:

STIMULUS	
----------	--

The Save Waveform DB dialog box appears.

b. Select the "results" waveform database and fill out the remainder of the dialog box as follows, then **OK** the dialog box:

	Save Waveform DB							
Waveform DB								
forces (Fo	rce Target)							
mywdb								
results (De	efault)							
Viewpoir	ıt							
Leafname	results_lab2							
File type	WDB Logfile Forcefile							
Replace								
ОК	Reset Cancel Help							

You could also save the results in forcefile or logfile format by clicking on the appropriate button. The "results" waveform database is saved to disk as indicated in the message area. 21. Exit QuickSim II by issuing the following:

(any window) Exit
-------------------

The Exit QuickSim II dialog box is displayed giving you one last chance to save information.

22. Verify that "Without saving" is selected, then click on the **OK** button:

Exit Q	uickSim
After Saving	Without saving
OK Re:	set Cancel

No further objects are saved and QuickSim II exits.

- 23. Clean up the remaining processes by performing these steps:
  - a. Close the QuickSim II Terminal transcript pad that remains.
  - b. Close the Design Manager session before logging out.



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 2 Solutions" on page A-3.

Then continue on to the next module.

#### **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 2 Solutions" on page A-3.

- 1. What form must stimulus take before it is issued to the kernel?
- 2. What are the benefits of storing waveform information in binary form vs. ASCII form.
  - a. \_\_\_\_\_\_ b. \_\_\_\_\_
- 3. What two ASCII formats can waveform database output be converted to/from?
  - a. \_\_\_\_\_
  - b. \_\_\_\_\_
- 4. When you issue the Force command, into which waveform database is the force placed, by default? \_\_\_\_\_\_
- 5. What menu item do you use to load a forcefile into a waveform database?

## **Module Summary**

This module contains information on how stimulus (event states) for design pins, nets, and buses is handled by QuickSim II (SimView/UI). More advanced procedures can be found in the *SimView Common Simulation User's Manual*.

- Waveforms. Contain an event stream that can either be connected to a signal as stimulus or kept for a signal as results. The concept of keeping an event stream as results is only available in SimView/UI-based simulators such as QuickSim II. Waveforms reside within waveform databases.
- Waveform databases. Contain one or more waveforms. Waveform databases maintain waveforms in a compiled form that is efficient for the kernel. Waveform databases can be created from forcefiles or logfiles, and can be written to forcefiles and logfiles. A waveform database must be "loaded" to be available for use. The "default" waveform database is "results" and the default "force target" waveform database is "forces" in a QuickSim II session. Any waveform database can be saved to disk, either to the design viewpoint or other location, and loaded into memory from disk. Several waveform databases can be merged as stimulus.

A distinction is made between stimulus and results waveforms.

- Stimulus waveforms are containers for existing event streams that can be connected to a design and that can be viewed in the Trace or List windows before or after a simulation run.
- Results waveforms reflect event streams created due to a simulation run and that can be viewed in the Trace, List, Monitor, or Flag Monitor windows only after a simulation run has completed. Each results event stream is associated with a pin, net, or bus

# Module 3 QuickSim II Design Invocation

Overview	3-2
Lesson	3-3
Design Manager Tool Invocation	3-4
Invoking QuickSim II from the Shell	3-6
Electronic Designs	3-8
Design Viewpoints and QuickSim II	3-10
Design Viewpoint Creation	3-12
DVE Invocation	3-14
DVE Default Windows	3-16
Setting Up a Viewpoint for QuickSim II	3-18
Design Configuration Window	3-20
Resolving Property Values	3-22
DVE Design Checks	3-24
Using TimeBase	3-26
Reporting Timing Information	3-28
Lab Preview	3-30
Lab Exercise	3-32
Procedure 1: Creating a Viewpoint	3-32
Procedure 2: Invoking QuickSim II	3-40
Test Your Knowledge	3-46
Module Summary	3-47

#### **Overview**



#### Lesson

Upon completion of this module, you will be able to:

- Describe a Design Viewpoint Object and its contents.
- Invoke the Design Viewpoint Editor directly on the new design (a design that does not contain a QuickSim II viewpoint).
- Specify the design configuration rules for QuickSim II using the menu item.
- Save the viewpoint for later use in QuickSim II.
- Exit the Design Viewpoint Editor.
- Invoke QuickSim II using the Design Manager Tools window method.
- Invoke QuickSim II from the shell using shell invocation options.



You should allow approximately 1 hour and 25 minutes to complete this module.

### **Design Manager Tool Invocation**

#### Double-click on QuickSim II Tools icon.

	QuickSim II
Design	Navigator
Symbol	Interface
Timing r	node Previous Unit Delay Constraint
Detail of	'Delay' timing mode Hidden Visible
Tim	ning mode Min Typ Max Unit
I	Use Full Linear Delays Delay Scale 1
Cor	nstraint mode Off State only Messages
Spi	ke model X immediate Suppress
	Hazard check Spike check:
	Contention check Suppress
	Model messages 🔄 X
	Toggle check Transport
	Simulator resolution 0.1 ns
L	_ Transport _ Blm check _ Blm debug
	OK Reset Cancel

### **Design Manager Tool Invocation**

When you open or double-click on the QuickSim II icon, the QuickSim II dialog box is presented to allow you to configure the invocation. It is here you specify the invocation options that correspond to switch options for the **quicksim** shell command. Here is a brief description of available options.

- **Design pathname** (required).
- Symbol | Interface. Allows using alternate symbol or interface.
- Timing Mode.
  - Previous. Allows you to use a saved state or setup object.
  - Unit. Sets Timing mode to unit delay.
  - Delay. Sets Timing mode to Typ delay mode. Make the detail visible to specify Min or Max timing mode.
  - Constraint. Turns on constraint checking and messages, sets timing to Typ, sets spike model to X\_immediate, and turns on all checking.
- **Detail of timing mode**. Allows you to see and/or change the additional invocation options available as follows:
  - Timing mode. Set the delay value to one of the triplicate values.
  - Delay scale. Multiplier used to adjust all timing values.
  - Constraint mode. Enables constraint checking and message reporting.
  - Spike model. Choose X-immediate (outputs X) or suppress on spike.
  - Spike check. Enables spike checking.
  - Hazard check. Enables checking for zero-delay loops in design.
  - Contention check. Enables reporting multiple drivers on a pin or net. The contention model must be set within QuickSim II.
  - o QuickPart messages. Reports these messages when they occur.
  - Toggle check. Keep number of circuit transitions at each connection.
- **Simulator resolution**. Adjusts the timestep of the event wheel in the simulation kernel. This value can't be changed within QuickSim II.



For more information on invoking an application from Design Manager, refer to the *Design Manager User's Manual*.

# Invoking QuickSim II from the Shell

Option Name	Abbrev	Option argument
component_name		
design_viewpoint		
-help		
-usage		
-I -S		root_name
-nodisplay	-nod	(default) -display
-timing_mode	-tim	min   max   typ   <b>unit</b>   lmin   ltyp   lmax
-delay_scale	-ds	scale_factor (1.0)
-time_scale	-ts	scale_factor (0.1ns)
-constraint_mode	-consm	off   state_only   message
-contention_check	-coc	off   on
-spike_check	-spc	off   on
-spike_warnings	-spw	<b>off</b>   s   x   t   sx   st   xt   all
-model_messages	-mm	off   on
-blm_check	-blmc	<b>off</b>   on
-toggle_check	-toc	off   on
-hazard_check	-hac	<b>off</b>   on
-dbg_blm		(defaultnone)
-spike_model	-spm	suppress   x_immediate
-delay_mode	-delm	inertial   transport
-setup	-set	setup_name (defaultnone)
-restore	-res	<pre>save_state_obj (defaultnone)</pre>
-absfile	-abs	abstract_signal_file (defaultnone)
-display		x_display (defaultcurrent display)
-geometry		x_geometry (MGC default)

## Invoking QuickSim II from the Shell

QuickSim II has defined several invocation options (switches) that allow flexibility in bringing up a simulation from a shell. This is not a complete description of switch options, merely a list to highlight frequently used options. They are explained here:

- -I|-S. Specifies a component interface (-I) or symbol (-S) for design root. Default: default interface specified in the root component interface.
- -NODisplay. This switch causes the simulator to invoke in the shell background. This is how you start a batch simulation.
- **-Delay\_Scale**. Provides a modifying delay (*scale\_factor*) for all timing values in the simulation. It can be used to create min or max timing from typical values.
- -CONStraint\_Mode. Enables constraint checking (*on*) or allows checking and message reporting (*messages*). Default: off.
- -?\_Check. Where "?" is the unique type of check. The quicksim command allows individual checks to be turned on or off upon invocation. Default: off.
- -SETup. Specifies a design object (setup\_name) to use to set up the simulator. The object configures the kernel upon invocation.
- **-REStore**. Specifies a *save\_state\_obj* used to restore a previous simulation state upon invocation.



Refer to the *Digital Simulators Reference Manual* for more on the quicksim command and invoke options.

# **Electronic Designs**



**Electronic Design** 

- Component. Object that contains models
  - Model types
    - Functional -- simulation model behavior
    - Graphical -- symbol
    - Technology -- timing models
  - Register and label the models -- "part"
  - Created with EDA application (Design Architect)
- *Viewpoint*. Configuration object -- Stores primitives, parameters, annotation connection
- QuickSim II requires both a *component* and a viewpoint

#### **Electronic Designs**

An electronic design (or just a design) is created with Electronic Design Automation (EDA) applications, primarily Design Architect. A design represents the circuitry of an electronic device, and can be as simple as a logic gate or as complex as an entire system. For use in QuickSim II, a design must include both a *component* and at least one *configuration*. The configuration is stored in a database object called a *design viewpoint*. The figure on the preceding page represents an electronic design.

A component is an object that contains a set of *models*. Each model describes either the functional, graphical, or technology aspects of the design. When the component is created, you register and label the models according to how you want them associated. The simulator selects the models during simulation according to how you associated them.

You can create the technology and timing using models called *Technology Files*, where you can build sophisticated pin-to-pin timing delays and dependencies. The Technology file source must be compiled and registered to be used by a component.

Creating the models for the component's graphical representation is done through the Symbol Editor in the Design Architect. You can create many graphical representations for the same component, to represent different graphical standards or logic representations.



For more information about registering and labeling models, refer to "Registration and Labeling" in the *Design Architect User's Manual*.

# **Design Viewpoints and QuickSim II**



Design viewpoint (dvpt):

- Defines set of configuration rules
- Manages back annotation (ba) objects
- Container where persistent (saved to disk) simulation information is managed
  - Setup object (quicksim\_setup)
  - Waveform databases (forces, results)
  - Save state object (quicksim\_state)
  - Timing cache (timing.data)

## **Design Viewpoints and QuickSim II**

To the simulator, the design viewpoint does two things: it defines the set of configuration rules (parameter, primitive, substitute, and visible property) that the simulator uses to evaluate the design, and it serves as a container where information related to the simulation can be stored, retrieved, and managed.

The simulator can store the following related objects inside the design viewpoint container:

- Save state object. Contains a complete kernel state at a specific point in simulation time. You can restore a saved state to return the simulation to the point at which you saved the state.
- **Timing cache.** Used only for full simulation timing; it is not used for unit delay timing. Once created, it is reused as long as the design viewpoint remains unchanged. When changes are made within the simulator, the affected timing information is recalculated and appended to the timing cache.
- Setup object. Contains simulator setup conditions that you can restore. Setup conditions include the kernel setup (timing and delay mode, design and hierarchical checking modes and settings, and BLM checking), the keep list, the run setup, a list of breakpoint settings, window positions and sizes, bus names, synonyms, expressions, and user-loaded userware.
- Waveform databases. Contain binary waveform data generated from the simulation or through stimulus input. The icons named "results" and "forces" are waveform databases. "forces" is the default name QuickSim II uses for stimulus force waveform database. "results" is the default name for the output of the simulation run.



# **Design Viewpoint Creation**

- Auto-tool mode -- QuickSim II creates a default viewpoint, if none exists, or uses existing one
- Batch mode -- Shell script (DVE) creates viewpoint tailored to company or design process
- Interactive mode -- You invoke DVE and create or modify viewpoint interactively

## **Design Viewpoint Creation**

All designs the QuickSim II accesses must have a design viewpoint. Certain configuration information contained in the viewpoint is required by QuickSim II. Other viewpoint is important for proper design evaluation.

There are several ways you can create a design viewpoint for your design, as indicated in the figure on the previous page:

• Automatic (tool generated) mode. QuickSim II will automatically generate a viewpoint if you have not previously created one. It uses a minimum set of configuration rules to configure your design and performs a design evaluation based on these rules. Assuming that the rules are valid and complete for your design, you are ready to simulate.

You should not use this mode if your design requires special rules or needs additional configuration information. Many ASIC design libraries require some parameters and properties to be defined in the viewpoint for the library to evaluate correctly. Using the automatic mode will leave many evaluations with incomplete (null) results.

- **Interactive mode.** You invoke DVE on your design and use the configuration menu items to create and customize a unique viewpoint. You would use this method to create a viewpoint name other than "default." You might also need to add parameter or primitive declarations that are unique to your design. You also use this mode to manipulate back annotation objects.
- **Batch** (script) mode. To set up a script, you invoke DVE on a typical or test design and use interactive mode to configure your design the way all designs should be configured. Add all unique property names and parameter values. Save the design viewpoint and exit DVE.

Now copy the transcript of this DVE session to an edit pad and clean up any extraneous functions and errors. Add the invocation commands to the script and enclose the DVE transcript as a "here document". Save the script to a executable location and test.

## **DVE Invocation**

You can invoke DVE on:

- A component
  - DVE looks for viewpoint named "default".
  - Creates it, if not found.
  - A design viewpoint
    - DVE opened on "named" design viewpoint.
    - Does not create it if name doesn't exist.
  - Stand-alone
    - Use File > Open > Design Viewpoint

Invoke from shell: **\$MGC\_HOME/bin/DVE** 

Invoke from Design Manager:

- a. Select a design or vpt, then Open > DVE
- b. Double-click on tool icon (stand-alone)

Open Design Viewpoint									
Component Name	Navigator								
Viewpoint Name default	Open as:								
Options NO YES									
OK Reset Cano	el								

## **DVE Invocation**

There are three modes you can use when you invoke the Design Viewpoint Editor. You can invoke DVE on:

- A component. When you specify the name of the component, DVE looks for the design viewpoint named "default". If the "default" design viewpoint exists, it is opened. If it does not exist, an empty "default" design viewpoint is created. You then use DVE to specify rules for the empty design viewpoint.
- A design viewpoint. When you invoke QuickSim II on a design viewpoint, DVE is opened on that design viewpoint. DVE does not create a new design viewpoint during invocation on a specific design viewpoint.
- **Stand-alone**. Once invoked, specify a component and the design viewpoint name. Use the **File > Open > Design Viewpoint** menu item or Open Design Viewpoint command to specify the name of the component and design viewpoint.

There are two environments from which you can invoke DVE, the Design Manager, and the shell. To invoke DVE from the shell level on a component, use the **dve** shell command specifying a component, a viewpoint, or stand-alone:

```
shell> $MGC_HOME/bin/dve /tmp/test.ckt
shell> $MGC_HOME/bin/dve /tmp/test.ckt/simulation.vpt
shell> $MGC_HOME/bin/dve
```

To invoke DVE on a component or design viewpoint in the Design Manager:

- Select the component or design viewpoint icon
- (Iconic Navigator window) > Open > DVE

To invoke DVE stand-alone in the Design Manager:

- Open the Tools window.
- Double-click on the DVE tool icon.

Note that you can also create a script that invokes DVE in batch mode, creating a design viewpoint, manipulating back annotations, checks the design, and then exits. This is especially useful if your site uses a common but custom viewpoint.

### **DVE Default Windows**

#### If you invoke DVE on a new viewpoint, you see:



#### **Design Viewpoint window -- Annotation objects**

#### **Design Configuration window -- Rules**

#### **DVE Default Windows**

When you open a design viewpoint, you get two default windows within the Design Viewpoint Editor session window. These are the Design Viewpoint window and the Design Configuration window. A brief description of these windows follows:

**Design Viewpoint window**. This window lists all of the objects related to the design viewpoint. Back annotation objects will be listed in this window. You can use this window to select the object that you wish to use. For example, if this window contains three back annotation objects named larry.ba, moe.ba, and curly.ba, then you can click on the "moe.ba" text to select and then open it.

**Design Configuration window**. This window lists the design configuration rules that you have applied to your design. These rules fall into four categories: parameter, primitive, substitute, and visible property. The meaning of these rules is explained on page 3-21.

The Setup menu has an item for each Mentor Graphics application that you can configure for. Executing the Setup menu for an application item issues groups of functions that set the default rules for that application. As you will see later, there are also individual functions that allow you to add single configuration rules or to modify existing rules.

# Setting Up a Viewpoint for QuickSim II

#### To set up design configuration for QuickSim II:

Setup > (Quick)SIM, Fault, Path, and Grade



The menu lists some of the other configurations that you can set up in the viewpoint.

# Setting Up a Viewpoint for QuickSim II

The process to set up the design configuration for a simulation within DVE is as follows:

- 1. Specify default configuration for QuickSim II using Setup > (Quick)SIM, Fault, Path, and Grade menu item. It provides the following:
  - Adds the following visible properties to the design viewpoint:

NOFAULT	MODELCODE	PINTYPE
MODEL	RISE	INIT
TIMEFILE	FALL	DECAY
MODELFILE	DRIVE	DTIME

• Defines the following Model property values as being primitives:

INV	PRXFER	PLA
BUF	CXFER	FPLD_MODEL
AND	CRXFER	LATCH
OR	WOR	REG
NAND	SWOR	\$GEN_QPT
NOR	CSWOR	\$QPT
XOR	NMOS	\$G5
XNOR	PMOS	\$HML
DEL	CMOS	\$BLM
RES	NSW	\$LM
NULL	CSW	\$MTM
XFER	PSW	SWIFT
RXFER	RAM	BRES
PXFER	ROM	

2. To add other parameter, primitive definitions, substitutions, or insertions to the viewpoint using the **Edit** > menu items. The next pages outline this process.

# **Design Configuration Window**

**Design Configuration rules** 

- Parameters -- value for a variable
- Primitives -- lowest hierarchical level for models
- Substitutes -- substituted for another property's value
- Visible properties -- visible to DFI and netlisters

		Design Co	onfigu	ration			-	
PARAMETER								$\Delta$
PRIMITIVE								
model	(	INV, BUF, ANI	D,OR,I	NAND, N	IOR,	XOI	R,XNOR,D	
SUBSTITUTE								
VISIBLE PROP	ΈF	RTY						
dtime	(	,	,	net,	,	)	-UPCASE	
decay	(	,	,	net,	,	)	-UPCASE	
init	(	,	,	net,	,	)	-UPCASE	
pintype	(	,	pin,	,	,	)	-UPCASE	
drive	(	,	pin,	,	,	)	-UPCASE	
fall	(	,	pin,	,	,	)	-UPCASE	
rise	(	,	pin,	,	,	)	-UPCASE	
modelcode	(	instance,	,	,	,	)		
modelfile	(	instance,	,	,	,	)		
timefile	(	instance,	,	,	,	)		
model	(	instance,	,	,	,	)	-UPCASE	
nofault	(	instance,	pin,	net,	,	)	-UPCASE	$\nabla$
$\bigtriangledown$								>

## **Design Configuration Window**

A design configuration is the part of the viewpoint that contains rules for evaluating the associated component. These rules are called *design configuration rules*. You can set or change four types of design configuration rules in a design viewpoint:

- **Parameters**. A *parameter* is a named value for a variable that may be resolved through a temporary value in Design Architect, a parameter rule in the design viewpoint, or through a technology file. Parameters may be declared as part of the design configuration process and are used to define values for variables in property values. An example of a parameter is using the name "technology" as the value for the model property and giving it a type expression. When the design viewpoint is created, you may specify a real value like ".8mic\_qpt" in the parameter rule, which the simulator then substitutes for "technology."
- **Primitives**. The component evaluation progresses from highest hierarchical level to a stopping point recognized by finding certain property names and values on instances. Any instance which is a termination point is called a *primitive*. To set which instances are to be considered primitive during evaluation, property names and optionally values, are listed in the primitive rule.
- **Substitutes**. The substitution rule specifies one or more property names whose values are to be substituted for another property's value. The substitution rule can also be used to temporarily change property values so that you can complete the evaluation.
- Visible properties. The visible property rule informs DVE which properties are visible to netlisters that use DFI and Netlist Module.

## **Resolving Property Values**

QuickSim II uses the order shown below to evaluate expressions and satisfy property values.



# **Resolving Property Values**

When the design is evaluated in the context of a design viewpoint and the system finds an undefined parameter or variable in an expression, it starts a search up through the design tree to find a value for that variable. The figure on the facing page illustrates the search path the system uses to find the value. As soon as a valid value is found, the search stops. The search is described as follows:

- 1. The property owner (net, instance pin, or instance body) is search first.
- 2. If the owner is an instance pin, the body properties of the attached instance are searched next. (If the owner is a net, this step is skipped.)
- 3. The body property list of the instance's component interface table is searched next. (If the owner is a net, this step is skipped.)
- 4. Does the design have more levels of hierarchy? If yes, the search moves up to the parent instance. The properties on the instance body are checked first, then the instance's component interface table is searched next.

At any time, the value of a parameter may be overridden by a back annotation specified in a connected back annotation object. If more than one back annotation object is connected, the BA objects are search in prioritized order.

- 5. After the parent instance on the top sheet is searched, the design viewpoint Parameters list is searched.
- 6. If a value is still not found, a corresponding technology file (if registered to the top-level component) and then the library data technology file (if registered) is searched for the parameter. If the value is still not found or if the technology files do not exist, an error is issued.
  - To use values in technology files, refer to "Understanding the Scoping Rules" in the *Technology File Development Manual*. For information on property resolution, refer to the "Rules for Resolving Property Value Variables" section in the *Design Architect User's Manual*.



#### **DVE Design Checks**

#### Miscellaneous > Check Design > Check Options

Check Design		
Display in Window? No Yes	Warning Checks?	
Write to Transcript? No Yes	All Warnings Synthesis No Warnings	
Write to File? No Yes	Simulation Checks? No Yes	
Electrical Rule Checks? No Yes Name Checks? No Yes		
Check Design Subtree? No Yes Expand Messages? No Yes		
OK Reset Cancel		

### **DVE Design Checks**

You can use DVE to check your design in the context of the currently defined design configuration rules before proceeding to your target tool. This is called *configured design checking*. DVE can perform extensive checks of the entire design hierarchy using rules in the design configuration and your target tool.

These checks also examine the design for mismatched connections, unique names, and parameter values. DVE uses the **\$check\_design()** function to call the Design Syntax Checker (DSC) functionality. The checks are also performed during invocation of the QuickSim II application.

You can examine the sheet that caused an error or warning by selecting the instance pathname in the Check Design window, and using the Open Selected command. Once you know which sheet caused the error, open Design Architect (leaving the DVE session open). Make the necessary changes to the sheets that contain errors and warnings, then save the component. You may also have errors and warnings that can be fixed by adjusting the design configuration rules or back annotations in DVE.

You can check the syntax of an entire design with respect to the current design configuration rules, using the **Miscellaneous** > **Check Design** menu item in the following procedure:

- Open the desired design viewpoint through invocation or use of the DVE Session > Open Design Viewpoint popup menu item.
- 2. Verify that the design configuration rules are correctly defined for the QuickSim II application.
- 3. Choose the Miscellaneous > Check Design > Check Options menu item. When the Check Design dialog box appears click on YES for "Simulation Checks?" and OK.
- 4. A Design Syntax Messages window opens and lists the messages as the checking progresses.

## Using TimeBase



TimeBase:

- Performs timing calculations
- Verifies rules in design context
- Stores resulting timing data for use in simulation

#### To invoke TimeBase: \$MGC\_HOME/bin/timebase <design> [switches]

#### If Missing a model statement:

// Error: Model property not found on instance '/'.

// NULL model inserted. (from: Analysis/Digital...

#### The -DEBUG switch:

- Presents a debug menu
- Does not store persistent timing results

## Using TimeBase

The TimeBase tool performs timing calculations and stores the resulting timing data for use in simulation. Its use is optional. After you are satisfied that each of the individual parts and its associated technology files are correct and operational, you can use the TimeBase command to verify the operation of the parts in the context of a design. When you invoke TimeBase on a design, it calls on each technology file within the design and evaluates timing equations, applying evaluation rules as necessary. These rules attempt to fill equation values of each technology file within the design hierarchy.

QuickSim II automatically uses TimeBase during invocation to build a *timing cache* of evaluated timing information. This process extends the invocation time significantly on large designs. If you have a valid timing cache saved from either a manual TimeBase build, or a previous QuickSim II run, it will be used.

To invoke TimeBase and its options, issue the command at the shell prompt:

shell> \$MGC\_HOME/bin/timebase <circuit\_pathname> [switches]

If problems exist, TimeBase provides error messages. For example, if you are missing a **model** statement, you will receive the following message.

// Error: Model property not found on instance '/'.
// NULL model inserted. (from: Analysis/Digital Sim Utilities

You then add the model statement to your Technology File, or add the model property to the instance, recompile, and run TimeBase again.

The -Debug switch for TimeBase allows you to enter the debugging mode to examine items such as equation calculation results and scoping paths. If you use the Debug switch (shown below), it stores no persistent timing results.

shell> \$MGC\_HOME/bin/timebase my\_design -Debug



For a complete listing of syntax and a description of timebase switches, refer to the *Technology File Development Manual*.

## **Reporting Timing Information**

#### **Report > Timing Cache (TimeBase Statistics)**

Info Messages	
********** Timing cache information **********	$\square$
Timing cache evaluation = 11.6667 seconds	
Timing cache size = 483365 Bytes	
Timing function calls = $20$	
Instances updated = 166	
Equation complexity metric = 2	

#### **Report > Timing**

Report Timing Info	
On Selected objects Named objects	
■ Kernel □ Evaluated □ Source	
OK Reset Cancel Help	

- Source--technology file equations
- Evaluated--calculated technology file equations Doesn't show timing mode or delay scale
- Kernel--actual timing values the kernel uses Evaluated timing modified by the scale factor

## **Reporting Timing Information**

When you invoke QuickSim II in one of the timing modes, TimeBase compiles the timing information, if it does not already exist. You can also use TimeBase to compile your timing prior to invoking QuickSim II. In either case, a timing cache is prepared, which contains timing information and statistics about your design.

You can report information on the timing build process using the **Report** > **Timing Cache** menu item. The information that is presented give you an idea of the size of the cache, and the time required to build it. This information can be useful in determining memory requirements and runtime performance. The Info Messages window on the previous page shows a typical report.

You can also select an instance in your design and report timing on it using the **Report > Timing** menu item. A dialog box appears with three choices for the type of timing information:

- **Source.** Displays the unevaluated source technology file information. This shows technology file statements
- **Evaluated.** Displays the calculated values from the timing equations in the technology file. This takes into account the timing mode (min, typ, max), but does not incorporate the delay scale.
- **Kernel.** Displays the actual timing values the kernel uses during simulation. This is the evaluated time for the specific timing mode, modified by the scale factor.

The desired information is displayed in the Timing Info window.

When troubleshooting timing, report kernel information first, to determine what pin or path caused the delay. Then view the source to determine the equations that produced the delay. This is the effect-to-cause approach.

# Lab Preview



- Invoke DVE standalone
  - Open an existing viewpoint
  - Delete configuration rules
  - Create default QuickSim II configuration rules
  - Add a parameter
  - Add a property substitute
  - Save the design viewpoint
- Invoke QuickSim II on the add\_det design:
  - Specify viewpoint objects
  - Report object information on an instance
  - Report timing cache statistics
## Lab Preview

In the lab exercise for this module, you will:

- Invoke DVE in standalone mode, without a design or design viewpoint specified.
- Open an existing design viewpoint to specify viewpoint objects and select a different model using DVE.
- Add a new property to an output pin that contains an expression.
- Define a parameter value for the expression.
- Invoke QuickSim II from the Design Manager using Tool invocation options.
- Invoke QuickSim II from the shell using shell invocation options.

## Lab Exercise

Note

If you are reading this workbook online, you might want to print the lab exercises to have them handy when you are at your workstation.

### **Procedure 1: Creating a Viewpoint**

- 1. Log into your workstation and set your current directory to your training directory.
- 2. Invoke the Design Viewpoint Editor (DVE) in stand-alone mode as follows:

shell> \$MGC\_HOME/bin/dve

The Design Viewpoint editor session window appears in default mode:

					DV	E					
MGC	File	Edit	<u>S</u> etup	<u>R</u> epoi	rt <u>M</u> i	scellane	eous	Help			
									SETU	Setup	
									DEBU	G	
										ECT SE	
											_ PRIM S PROP <b>UPVPT</b>
									ADD PARA	M PF	
F1	F2 Unselect Al	F3 Open Dvpt Close Dvpt	F4 opup Menuer	F5 port Obje dd	F6 Propert ange Pro S	F7	F8 Open Shee pen Select	F9	F10 ulldown Me	F11 Read File Edit File	F12 Pop Windo lose Windo
eopen Sele		Save Dvpt	Reselect he	eck Desigele	te Prope C	Open Up	Open Dow				

- 3. Open a viewpoint on the LATCH design by doing the following steps:
  - a. Click on: SETUP

The Open Design Viewpoint dialog box appears.

- b. Click on the Navigator... button, which brings up the list navigator.
- c. Locate and select *\$HOME/training/qsim\_n/LATCH* component.
- d. Select the LATCH component and **OK** the navigator.

This places the path in the dialog box. Note that the "Viewpoint Name" field already contains the name "default". This is the name of the viewpoint QuickSim II created in Module 1 when you saved the Lab Exercise. The dialog box should be filled out as follows:

Open Design Viewpoint						
Component Name/training/qs	sim_n/LATCH Navigator					
Viewpoint Name default	Open as: Editable Read Only					
Options NO YES						
OK Reset Cancel						

e. Now **OK** the dialog box.

Two windows appear: the Design Viewpoint window and the Design Configuration window. The Design Viewpoint window title gives the path to the design object. (List here:)

The Design Configuration window lists the rules that are currently set for your LATCH design. The rules shown are the default rules that were used when you invoked QuickSim II on this design in Module 1, and then were saved with the viewpoint.

- 4. Delete the configuration rules as follows:
  - a. Place the graphical pointer on the line below "PRIMITIVE" in the window and click the Select mouse button.

The line is selected (highlighted) as shown:

PRIMITIVE							
model	(INV,	BUF,	AND,	OR,	NAND,	NOR,	XOR,
SUBSTITUTE							

b. Choose: (**Design Configuration popup**) > **Delete** 

The line disappears from the configuration.

c. Choose: (Design Configuration popup) > Select > All Visible Properties

This is an easy way to select all the visible properties that are in the configuration.

VISIBLE PROPE	RTY						
nofault	(inst	ance,	pin,	net,	,	)	
model	(inst	ance,	,	,	,	)	
timefile	(inst	ance,	,	,	,	)	
modelfile	(inst	ance,	,	,	,	)	
modelcode	(inst	ance,	,	,	,	)	
rise	(	,	pin,	,	,	)	
fall	(	,	pin,	,	,	)	
drive	(	,	pin,	,	,	)	
pintype	(	,	pin,	,	,	)	
init	(	,	,	net,	,	)	
decay	(	,	,	net,	,	)	
dtime	(	,	,	net,	,	)	$\overline{\langle}$

d. Choose: (**Design Configuration popup**) > **Delete** 

The visible property configuration rules are also deleted. You should now have no configuration rules defined. Let's define some.

5. Show the Transcript window in the vacant area of the session window as follows:

#### Choose: MGC > Transcript > Show transcript

Use window manipulation techniques to position the transcript at the right side of the session area so that it does not cover the other windows.

6. Set the design configuration rules for QuickSim II using the following menu choice:



If the Transcript window is still active, you will not see the setup menu item in the menu bar. You must first make one of the other windows active.

#### Choose: (Menu bar) > Setup > (Quick)SIM, Fault, Path and Grade

Notice what happens in the Design Configuration window. The list of configuration rules for QuickSim II are added to this window.

Notice also the entries in the transcript window. You can issue the "add..." functions independently in DVE to create these objects. This is how most ASIC vendors develop viewpoint creation scripts.

Name the function that created the "model (INV, BUF...." entry.

Function Name = \_\_\_\_\_

- 7. Add a substitute to the list for the QBRISE property as follows:
  - a. Choose: (Design Configuration popup) > Add > Substitute
  - b. When the dialog box appears, enter:

Substitute value of property QBRISE						
Substitute value of property QBRISE						
With value of property QRISE						
OK Reset Cancel						

c. Then **OK** the dialog box.

Note that the entry "qbrise" is in lowercase even though entered in uppercase. Property names are not case-sensitive.

d. Choose the same menu item again only this time enter:

Add Substitute						
Substitute value of prope	rty qbfall					
With value of property	qbfall					
With value of property	qfall					
OK Reset	Cancel					

e. **OK** the dialog box.

This command tells QuickSim II to substitute the value of qfall for qbfall only if a value for qbfall doesn't exist.

When does this rule take effect?

8. Add the value for the TCAP parameter as follows:

The TCAP parameter is used to add timing delays to a component based on capacitance loading. In this step, you will add a value that is used in placed of this parameter.

- a. Click on: **SETUP**
- ADD PARAM
- b. When the Add Parameter dialog box appears enter:



The parameter statement appears in the Design Configuration window.

9. Check the design against the design rules using the following:



The Design Syntax Messages window appears. This window itemizes each of the checks that are performed on the design, and lists the warnings and errors associated with each check. The following note should be displayed when the checks are complete:

**I** Design Syntax Checks completed - 0 status messages returned.

10. Check the design again using simulation checks as follows:

Note: Basic and warning syntax checks that you just performed do not check the design for simulation rules.

a. To perform simulation checks, choose:

#### Miscellaneous > Check Design > Check Options

The Check Design dialog box appears. It shows you which checks are performed in the default mode (as in the previous step).

b. Change "Simulation Checks?" and "Expand Messages" to Yes:

	Check Design					
Display in Window?	No Yes Warning Checks?					
Write to Transcript?	No Yes All Warnings Synthesis No Warnings					
Write to File?	No Yes Simulation Checks? No Yes					
	Electrical Rule Checks? No Yes Click here					
	Name Checks? No Yes					
Check Design Subtree? No Yes Expand Messages? No Yes						
OK Reset Cancel						

c. **OK** the dialog box.

A second (Design Syntax Messages#2) window appears (the window may be hidden behind the original window). Note that "Simulation syntax checks" were added.

d. Now close *both* of the Design Syntax Message windows.

11. Save the design viewpoint as follows:



The design viewpoint changes are saved to the viewpoint. Notice that the version is now incremented ("default\_2"), as shown in the title area of the Design Viewpoint window.

12. Exit DVE as follows:

Choose: (**DVE Window Menu** ) > <u>C</u>lose

If you had unsaved changes, a prompt bar would appear, giving you a second chance to save.

You have now created a custom viewpoint for the LATCH design. This viewpoint specifies the primitives, parameters, and properties that QuickSim II uses when you run a simulation.

## Procedure 2: Invoking QuickSim II

In this procedure, you will use invocation command options to configure QuickSim II kernel timing and checking. You will also examine how QuickSim II uses TimeBase to build timing for a design.

- 1. If necessary, log into your workstation and set your working directory to your *qsim\_n* directory.
- 2. Invoke QuickSim II on the "add\_det" design using the quicksim shell command and several switch options, as follows:

The beginning of this section explained the switches used with the quicksim shell command. You will use several of these switches to set up QuickSim II upon invocation. Be sure your current directory is set to your  $qsim_n$  directory before proceeding.

In a shell, issue the command:

What will be the environment for the following options after QuickSim II invokes? (For a list of QuickSim II shell invocation options, see page 3-6.)

- 1. Timing mode: \_\_\_\_\_
- 2. Constraint mode: \_\_\_\_\_
- 3. Simulator resolution:
- 4. Model messages: \_\_\_\_\_
- 5. Delay scale: \_\_\_\_\_
- 6. Spike model: \_\_\_\_\_
- 3. Maximize the QuickSim II session window.
- 4. Create the schematic view window using the palette button
- 5. Use the maximize button to maximize the schematic view window so that it fills the session. This method allows you to return this window to its original size in a later step.

- 6. Use the (View All) stroke to adjust the schematic view so the entire schematic fills the schematic view window.
- 7. Report information on the 74ls161a instance as follows:
  - a. Verify that all objects are unselected, then select the 74ls161a instance.
  - b. Click on: DBG GATES

S REPORT SELECTED

The Objects report window appears, containing information about the instance. Note that I\$9 is highlighted, since the instance is selected. This information is duplicated on the next page for reference.

- c. Use the Maximize button to enlarge the window so that all information is visible.
- d. Using this information, identify the following:
  - i. Simulation Model property value \_\_\_\_\_
  - ii. Simulation model type \_\_\_\_\_
  - iii. 74ls161a symbol ("Properties: type")
  - iv. 74ls161a component version \_\_\_\_\_
  - v. Simulation timing mode \_\_\_\_\_
  - vi. Spike reporting \_\_\_\_\_
  - vii. Hazard checking \_\_\_\_\_
- viii. State of the "B" input pin \_\_\_\_\_
- ix. State of the "RCO" output pin \_\_\_\_\_

Dbject		0	
1 Selected Instance /I\$9		2	$\overline{\square}$
Primitive Instance : TRUE Container Model Type : Schematic Sheet Sheet Name :add_det/schematic/s Sheet Version : 20 Defining Comp Name : path/qsim_n/lib/74ls1 Defining Comp Version : 1 Number of pins : 14 Properties: qp_prim : "TRUE" inst : "I\$9"	heet 61a	:1	
comp : "74LS161A" model : "\$G5" ref : "" type : "MG_STD" brd_loc : ""			
part_no : "" geom : "" pow_typ : "" Simulation object data: Model type : QP Spike model : suppress			
Constraint mode : Off Timing mode : min Delay scale : 1 Checks - hazard : Off spike reports contention : Off toggle unspecified path : Off stability	: A] : Of	Ll f	
Model messages : On stability stat Total pins : 14 IN: 9 OUT: 5 IO: 0 IXO: 0 INTERNA Dumping Information for QuickPart	e : L :	0	
INPUTS : A : OS, B : OS, C : OS, CLK : XR, D : OS OUTPUTS: QA : XS, QB : XS, QC : XS, QD : XS, RCO	, E1 : XS	1₽ 5,	

- 8. Do not close the Object report but pop it behind the schematic view window.
- 9. Report object timing as follows:
  - a. Choose: **Report > Timing**

The Report Timing Info dialog box appears, allowing you to choose the type of report detail you want. The "Kernel" item is the default detail.

b. Click on "Evaluated" and "Source" options also, as shown:

Report Timing Info						
On Selected objects Named objects						
Kernel Evaluated Source						
OK Reset Cancel Help						

c. **OK** the dialog box.

The Timing Info report window gives you information on all of the timing detail for the 74ls161A instance. The "Source" timing info is shown first. The second block of text shows the "Evaluated" info. The final area is the "Kernel" timing data, which incorporates scaling.

Since there is too much information to show, the following figure shows the comparable sections of each group of information. The "Source" information is under the label "Report Timing Info (Unevaluated Technology File)." The "Evaluated" information is listed beneath "Report Timing Info (Cached)." The "Kernel" information is listed below the label "Report Timing Info (Scaled)."

```
Timing Info
                                                           П
  Report Timing Info (Un-evaluated TechFile) Inst /I$9 of TYPE QP
  Technology File timing statements for Technology File:
  fMAX = 25 (MHz) ON CLK(AH)
  fMAX = 25 (MHz) ON CLK(AL)
  tP = 10, 20, 28 (ns) ON _CLR(AL) TO QA, QB, QC, QD(AL
... < information deleted > ...
  Report Timing Info (Cached) for Instance /I$9 of TYPE QP
  Pin Timing Info for Pin QA of PINTYPE OUTPUT
      RISE Delay: 0, 0, UNCALC (ns)
      FALL Delay: 0, 0, UNCALC (ns)
  Pin Timing Info for Pin QB of PINTYPE OUTPUT
      RISE Delay: 0, 0, UNCALC (ns)
      FALL Delay: 0, 0, UNCALC (ns)
  tP = 10, UNCALC (ns) ON _CLR(AL) TO QA(AL)
  tP = 10, UNCALC (ns) ON _CLR(AL) TO QB(AL)
  tP = 10, UNCALC (ns) ON _CLR(AL) TO QC(AL)
  tP = 10, UNCALC (ns) ON CLR(AL) TO QD(AL)
... < information deleted > ...
  Report Timing Info (Scaled) for Instance /I$9 of TYPE QP
  Pin Timing Info for Pin QA of PINTYPE OUTPUT
      Scaled RISE Delay: 0 (ns)
      Scaled FALL Delay: 0 (ns)
  Pin Timing Info for Pin QB of PINTYPE OUTPUT
      Scaled RISE Delay: 0 (ns)
      Scaled FALL Delay: 0 (ns)
  tP = 10 (ns) ON \_CLR(AL) TO QA(AL)
  tP = 10 (ns) ON CLR(AL) TO QB(AL)
  tP = 10 (ns) ON _CLR(AL) TO QC(AL)
  tP = 10 (ns) ON CLR(AL) TO QD(AL)
... < information deleted > ...
  tH = 3 (ns) ON ENP(L) TO CLK(AH)
      WITH ' CLR'(H)
  tH = 3 (ns) ON LOAD(V) TO CLK(AH)
      WITH ' CLR'(H)
  tW = 25 (ns) ON CLK(V)
  tW = 20 (ns) ON _CLR(V)
```

10. Report the timing statistics on the timing build process.

Choose: Report > Timing Cache

The Info Messages window appears containing the Timing cache information. The numbers shown may vary, depending on the workstation:

	Info Messages	
	********* Timing cache information **********	$\square$
	Timing cache evaluation = 0.15 seconds	
	Timing cache size = 66568 Bytes	
	Timing function calls = $0$	
	Instances updated = 14	
	Equation complexity metric = 0	$\overline{\neg}$
$   \leq  $		

- 11. Exit QuickSim II without saving anything.
  - a. Choose: (QuickSim II Window Menu) > <u>C</u>lose (In some window environments, this may be the "Quit" menu item).
  - b. When the dialog box appears, click on "Without Saving"
  - c. **OK** the dialog box.

QuickSim II exits.



### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 3 Solutions" on page A-5.

Then continue on to the next module.

## **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 3 Solutions" on page A-5.

- 1. Which QuickSim II invoke switch loads a setup object? \_\_\_\_\_
- 2. Define "design viewpoint": \_\_\_\_\_\_
- 3. If you have a property name that has no value, what DVE operations can you use to assign a value to that property?
- 4. What is the name of the default viewpoint created by QuickSim II or DVE if you do not specify a name? \_\_\_\_\_\_
- 5. List the DVE menu path that configures the viewpoint for QuickSim II.
- 6. How do you perform a design check within DVE?

## **Module Summary**

QuickSim II invoke options give you control over all kernel checking, environment setup, viewpoint assignment, and display mode. This control can be set up on and instance-by-instance basis, or can be set for the entire design. You control the timing, spike models, hazard check, contention model, scale timing, and messages. Use these options to enhance your batch-mode simulations.

A design viewpoint is incorporated to automatically attach objects to a design upon invocation. You can save your kernel setups, waveform databases, and property back annotations to this data object.

There are two methods of creating a design viewpoint. A default viewpoint is created if you invoke QuickSim II on a design with no viewpoint. Or you can invoke the Design Viewpoint Editor directly on a design and create a customized viewpoint using the DVE commands.

DVE allows you to customize the rules that apply to the design. These rules include:

- **Parameters**. Values to be use for design variables. Includes equations.
- **Primitives**. Sets the lowest level for simulation models.
- **Substitutes**. Allows one property's value to be substituted for another property's value. It can also provide a property value where none exists.
- Visible Properties. Informs DVE which properties are visible to DFI and netlisters that use the Netlist Module.

Finally, you learned that Design Viewing and Analysis Support (DVAS) provides selection, viewing, and grouping functions. With cross-window highlighting and source viewing, you can examine all relationships of objects within the design.

# Module 4 Simulation Models

Overview	4-2
Lesson	4-3
Database Structure	4-4
Model Selection	4-6
Component Interface Browser (CIB)	4-8
"Browsing" a Component Interface	4-10
Using the Component Window	4-12
Modeling Methods	4-14
Modeling Comparison	4-16
Schematic Models	4-18
Other Model Examples	4-20
Memory Table Models	4-22
Initializing RAMs and ROMs	4-24
How Models Are Registered	4-26
Verifying Models	4-28
Technology Files	4-30
Lab Preview	4-32
Lab Exercise	4-34
Procedure 1: Examining Models	4-34
Procedure 2: Verifying Models	4-44
Procedure 3: Changing a Technology File	4-48
Test Your Knowledge	4-52
Module Summary	4-53

## **Overview**



## Lesson

On completion of this module, you should be able to describe:

- Modeling techniques for QuickSim II simulation. This includes simulation primitive models, Schematic models, Behavioral Language models (BLMs), QuickPart models, QuickPart Table models, VHSIC Hardware Description Language (VHDL) models, and Hardware models.
- The component interface concept. This includes the part object and how it is used to reference models for a component.
- How to examine a component interface using the Component Interface Browser (CIB)
- Model registration. This includes how models are registered and what tools perform the registration.
- Technology files. You should be able to modify and compile a technology file so that it is registered with a specific interface.
- State tables and how they are used with QuickParts.
- How component modeling techniques provide design incrementality.



You should allow approximately 1 hour and 40 minutes to complete this module.

## **Database Structure**



- Component

   contains stuff
  - o type "component"
- Schematic

   holds sheets
- Part (interface)

   list of interfaces
  - pin/property map
  - model registry

- Sheet
  - $\circ$  symbol references
  - model references
- Symbol

   references part
- Models

   referenced by part

## **Database Structure**

The purpose of this figure is to show the structure of a simple design. Each object within the figure has a unique function in the design and simulation process. These objects may be files, filesets, directories, or combinations of these objects, to provide a single function. A list of these objects is shown below.

The arrows show the interdependency of these objects to each other. The arrows represent design *references*, with the direction of the arrow away from the owner of the reference. The validity of these references provide the integrity of the design. Following the object name is the fileset name attached to the object in the database.

- **dff** -- **mgc\_component**. A container that houses V8 Mentor Graphics component design objects (in this case, for the D-type flip-flop).
- **Symbol** -- **mgc\_symbol**. A fileset that contains the graphics, pins, and properties for the symbol used on Design Architect sheets.
- **Part** -- **Eddm\_part**. The fileset that contains the component interfaces for this component. Component interfaces identify the models that are available for this component
- Schematic -- mgc\_schematic. The schematic object includes a container named schematic that parents all of the sheets for this component.
- Sheet -- mgc\_sheet. Sheets created in Design Architect that comprise the schematic model.
- **Models**. Models are referenced by the component interface. The registration process places models in the interface, where the model is available for use.



For additional information on the component directory structure, refer to the *Digital Modeling Guide*.



- Model property -- identifies model
- Simulator compares Model property to labels

## **Model Selection**

When the analysis tools need to process an instance on a schematic sheet, the tools must select which registered model to use for the instance. This selection process is called *model selection*. The list of model values (\$schematic, \$G5, \$LM, \$BLM) for all of the instances is created when the design is evaluated, that is, when the in-memory or persistent (saved to disk) design viewpoint is created.

The instance appearing on a schematic sheet is simply the graphical representation of a particular component interface. Because the instance represents a particular component interface, the system can tie the instance to that specific component interface.

The Model property value for the instance is key to selecting which model the analysis tool chooses during model selection. The simulation tool's selection algorithms compare the value of the Model property against the labels of its registered models. The model(s) which have labels that match the Model property value then become part of the instance definition.



For additional information on model selection from within QuickSim II, refer to the *QuickSim II User's Manual*.

# **Component Interface Browser (CIB)**



• Examines the Component Interface (part object)

- **o View interfaces**
- Edit model labels
- **o Unregister models**
- Validate models

# **Component Interface Browser (CIB)**

The Component Interface Browser provides a quick and convenient way to inspect and change the contents of a component interface. CIB provides the following capabilities:

- View interfaces. View of all interfaces for the specified component.
- Edit model labels. Add a new label, alter an existing label, or delete a label
- Unregister models. Unregister a specific model from a component interface.
- Validate models. Verify net definitions in the registered model to pins.

CIB is a shell-level application only. To invoke:

```
shell> $MGC_HOME/bin/cib
shell> $MGC_HOME/bin/cib $HOME/training/qsim_n/lib/dff
```

You can open or change components without exiting CIB. To change components, enter the Open command:

```
CIB > open $HOME/designs/jkff
dff :: ansi > open $HOME/designs/jkff
```

The prompt changes to show the opened component and its default interface (ansi). To view the contents of all component interfaces, enter the View command:

jkff :: mg\_std > view

The active component can be closed using the Close command. This resets the prompt to "CIB >" on the command line.

jkff :: ansi > **close** CIB >

## "Browsing" a Component Interface

COMPONENT pullu	DEFAUL	I INTERFACE IS: pullup
INTERFACE: pu	llup	
PINS:		
Comj	piled User	
Id # Pin	Name Pin Nar	ne Properties
1 UP	UP	(pin, UP)
BODY PROPERT	IES: No Body Pr	roperties for this Interface
INTERFACE MO	DEL ENTRIES:	
Model Entry	Туре	Model Info
0	mgc_symbol	Path: /lib/pullup/pullup
		Labels: 'default_sym'
		Status: Valid for interface;
		Valid for property
1	mgc_schematic	Path: /lib/pullup/schematic
		Labels: '\$schematic'
		'schematic' 'default'
		Status: Valid for interface;
		Valid for property

### **Example of interface problems:**

Model Entry	Туре	Model Info
1	mgc_schematic	Path: /lib/dff.alt/schematic
		Labels: '\$schematic'
		'schematic' 'default' '\$G5'
		Status: NOT valid for interface;
		NOT valid for property
2	Technology	Path: /lib/dff.alt/dff.g5>
		TROUBLE FINDING/LOADING MODEL !!
Attempt to cor	nnect child Design o	<pre>object failed (child /lib/dff.alt/dff.g5)</pre>
Design object	/lib/dff.alt/part of	contains a reference to
/lib/dff.alt/d	lff.g5	
Attempt to ini	tialize object fai	led (name /lib/dff.alt/dff.g5)
Error opening	attribute file (nam	<pre>ne /lib/dff.alt/dff.g5.Tf_tfile_do.attr)</pre>
Open of File /	lib/dff.alt/dff.g5	.Tf_tfile_do.attr failed.
No such file o	or directory	
		Labels: '\$dff.g5.cmp'
		Status: Valid for interface;
		Valid for property

## "Browsing" a Component Interface

When you view the contents of a component interface using the Component Interface Browser, the data is formatted and dumped to the shell pad in text form. The opposite page shows a CIB view of the */user/gen\_lib/pullup* component.

The three parts of the interface are grouped as follows:

- **The name.** This names the interface. It is important to name the interface because many components have multiple interfaces within the same component name (*/user/gen\_lib/rip*, for example, shown on the next page).
- **The pin list.** This identifies all of the interface pins and properties associated with each pin.
- **The Model table.** The model table information determines which symbol is used as the graphical instance on a schematic sheet, and also which model is used for analysis. Each model table entry contains information about the type of model, pathname, labels, and validity.

As explained earlier, the label can be used as a value to the Model property to allow flexibility in using simulation models. A default model is assigned and used by different applications to avoid *null* models. QuickSim II will declare a model as null when a default cannot be found and a model has not specifically been declared.

0

It is beyond the scope of this training workbook to present all of the CIB capabilities. For more information on the Component Interface Browser, refer to the *Component Interface Browser User's and Reference Manual*.

# **Using the Component Window**

- Examine component interface graphically
- Much quicker than CIB
- Built into QuickSim II and SimView

### To open window, choose:

### MGC > Design Management >

### **Open Component Window**



## **Using the Component Window**

A Design Hierarchy window is available from the MGC > Design Management menu that allows you to explore your design structure and model interfaces. This window allows you to:

- Examine the component interface tree structure in graphical form. The same component icons used in the Design Manager iconic navigator are used in this window.
- Examine component part interface and registration information. You can access this information much more rapidly than with CIB, and you get the information directly in QuickSim II, rather than in another shell.

The Design Hierarchy window presents its information in two sub-windows:

- **Component Information**. This sub-window provides a tree structure of your design hierarchy, showing the design path of each object in the tree. By double-clicking on an object in the tree, you can reveal the level of hierarchy below that object.
- **Registered Model Info**. When you select interface text (italicized text objects just below the component) in the Component Information sub-window, The text information about its part interface is displayed in this window. Not all of the interface model information is presented at one time, but can be accessed.

To display additional information about a particular model, you can doubleclick on the model icon or pathname in the Registered Model Info window. This present a list of registered labels for this model.



Unlike CIB, the Component Window is a "Read Only" window. You must use CIB or the Reg\_model command to change component interface information.

## **Modeling Methods**

Model Type	lcon	Modeling Technique Used to Create Functional Models
Behavioral Language Model (BLM)	BLM	Compiled high-level language source file that describes a device.
Built-in Model	No Icon	Logical behavior is built into the QuickSim II analysis tool.
Logic Model (LM)		LM-Family of hardware modelers.
Memory Table Model (MTM)		State table file compiled for memory devices that control RAM/ROM function
QuickPart Schematic Model	QPS	Object file of QuickPart compiled schematic sheet.
QuickPart Table Model	QPT	Compiled ASCII files similar to logic tables.
Schematic Model		Created with Schematic Editor by connecting various models.
System-1076 (VHDL)		Object files compiled with the Design Architect VHDL Editor.

## **Modeling Methods**

You can use a variety of modeling methods together in a QuickSim II simulation. The table on the previous page lists the methods available for creating functional models, and the icon associated with each model type. The following are different modeling techniques that can be simulated in QuickSim II:

- Behavioral Language models (BLMs). Compiled code describes models.
- **Builtin models**. QuickSim II recognizes builtin models such as AND, OR and BUFFER functions. No function descriptions are required.
- LM-Family of hardware modelers. Real hardware devices are used.
- Memory Table Models (MTM). A table describes input and output signals.
- **QuickPart Schematic models**. Schematics and a timing file are compiled together to create a faster, smaller model.
- QuickPart Table models. Like MTMs, a state table describes functionality.
- Schematic models. Schematic sheets of other components and properties.
- **VHDL models (System-1076)**. A hardware description models the design at a high level. VHDL can be *synthesized* to gate level models.

When the Design Manager recognizes a design object identifier, it presents a unique icon for that object. Using operating system shell commands does not preserve this relationship and can corrupt the object relationships. The icons used for other objects can be found in Table B-1 on page B-1.



For general information on modeling techniques, refer to the *Digital Modeling Guide*. For detailed information on a modeling technique, refer to the appropriate modeling manual. For additional information on the fileset concept, fileset icons, or the Design Manager, refer to the *Design Manager User's Manual*.



# **Modeling Comparison**

## **Modeling Comparison**

The figure compares modeling methods as a function of level of abstraction. This is important to you because it can affect (improve) your simulation performance. For example, if gate-level models are used in complex board design with many VLSI components, memory requirements are high, and simulation performance suffers. Here is a brief description of the modeling methods available:

- **Builtin models**. Provide basic function for small designs. As gate count grows, simulation overhead increases to manage the analysis.
- **QuickPart models**. Provide compiled gate and timing functionality. Simulation throughput is increased because gate analysis is consolidated. QuickPart Tables provide functionality without the expense of gate count.
- **BLMs**. Achieve functionality and timing using a compiled structural language. Although large systems can be managed using BLMs, cost is high in the development and verification of the BLM.
- **Hardware Modeling**. Allows use of real devices (ICs) to model function (and optionally timing). This option requires equipment overhead to extract functionality from the device and provide to the software analysis.
- **System-1076 (VHDL)**. Describes a design from a behavioral system level to a structural gate level. This method can model devices with levels of abstraction that go beyond electrical limits. Some government specifications require design models using this method.

As a hardware design is refined, all of the modeling methods can be combined to perform full system simulation. If portions of your design have already been created with a modeling method such as a behavioral language model (BLM) or a hardware model, they can be simulated in QuickSim II concurrently with gates, or with System-1076, without consideration to modeling method.



For more information on digital modeling methods, refer to the *Digital Modeling Guide*.

## **Schematic Models**

Model label: \$schematic





- Robust registration capabilities

   Choose component interface to register schematic
   Use your own labeling scheme
- Multiple schematics under one component
  - Model selection changes Model property value to select the desired schematic
### **Schematic Models**

Schematic models have the following features:

- Robust registration capabilities
- Multiple schematics supported under the same component

By using schematic registration with component interfaces, you can have multiple schematics under the same component. The model selection algorithm works with the Model property value to select the desired schematic.

Schematics must be registered with the component interface. Each component can have multiple models and can have one or more schematics registered with it. The registration process assigns labels to each model so that a descriptor selection algorithm can determine which schematic to use during analysis of the circuit.

The registration scheme for schematics is very robust. You can pick and choose which component interfaces to register the schematic, and also use your own labeling scheme.



For details on Schematic models see the *Digital Modeling Guide* and the *Design Architect User's Manual*.

# **Other Model Examples**



### **Other Model Examples**

- A behavioral language model is a compiled high-level language source file that describes the behavior of a component. BLMs have the following capabilities:
  - Include files
  - 64-bit timing support
  - Technology file support
  - Entry point for updating BLMs
  - QuickSim II invocation switch (-BLM\_Check) provides error checking
- A QuickPart Schematic model is compiled from a schematic (sheets) and optionally a Technology File. Here is a list of QuickPart Schematic features:
  - Timing Optional--handled by the Technology Compiler
  - Asynchronous feedback loops are supported
  - Hierarchical modeling allowed
  - Bi-directional pins--model Pintype property of IXO.
- A QuickPart Table model is compiled from an ASCII state table functional description and a Technology File. This type of model offers:
  - Efficient workstation memory usage
  - Fast simulation evaluation
  - Easy to read and debug logic implementation
- System-1076 is a compiled VHDL code providing a software model that you can test using QuickSim II. System-1076 features:
  - o model and test a system at a high level of abstraction
  - o synthesis tool can create a gate-level implementation
  - o use VHDL to mimic the behavior of "off the shelf" parts
  - System-1076 is based on the IEEE Std 1076-1987
  - o integrated into Mentor Graphics IDEA Series

0

For details on how to build a Technology File, refer to the *Technology File Development Manual*. For information on how to use a technology file with a builtin model, see the *Digital Modeling Guide*. For details on builtin models see the *Digital Modeling Guide* and the *Digital Simulators Reference Manual*.



## **Memory Table Models**

Memory Table models consist of:

- Graphical description--symbol uses \$MTM model
- Functional description-- compiled ASCII file
  - File contains functional table of port's behavior
  - MTM command compiles and registers
- Data initialization file (optional)
- Technology description (optional)

### **Memory Table Models**

A Memory Table model (MTM) is the recommended way to implement common RAM and ROM design objects. An MTM uses a compiled state table to provide the functional description. This table provides the control logic for the memory device, while a second ASCII file provides initial memory content information.

Memory Table models cannot be used to model every type of memory device. Currently, this type of model supports static RAM or ROM devices having one or more ports. Other types of memory devices must be modeled by using a different type of digital model--generally a BLM. Memory Table models consist of:

- **Graphical description.** To create a Memory Table model, you must first provide a graphical description. This is a symbol that uses the \$MTM model property. You can add timing to the symbol instance.
- **Functional description.** The functional description of a Memory Table model is a compiled ASCII file that specifies the functionality of the device's pins and ports. This file contains a functional table that specifies each port's behavior. After you create this source file, you must compile and register the code. The MTM command performs both operations.
- Data initialization file (optional). The file uses the same format used for existing memory models used with QuickSim II and found in Mentor Graphics memory libraries. The figure on the previous page shows the initialization file "connected" to the blank box above the model table. This illustrates that the initialization file is "pointed to" by the value of the Modelfile property found in the property list.
- **Technology description (optional).** Technology information can be optionally registered with the interface, if needed. Timing information can also be added to the graphical instance.
- 0

For information on the format of the initialization file, refer to the *QuickSim II User's Manual*. For a complete description of the steps necessary to create a Memory Table model, refer to the *Memory Table Model Development Manual*.

## **Initializing RAMs and ROMs**

If any X's appear on the ROM address bus, the ROM outputs all X's on the data bus.

Modelfile property value--pathname to ASCII file:

- Data and addresses must be hexadecimal or X
- Specify memory data using the form:

   address / data ; or
   low\_address-high\_address / data ;
- Any letter can be in upper or lower case
- Underspecifying--simulator fills in with zeros 16-bit example: FF = 00FF

#### Example 1: Initializing RAMs or ROMs to zero:

```
# 16-bit input-output
0-FFFF / 0 ; # Put zeros in all memory locations
```

### Example 2: Using X in ROM or RAM modelfile:

```
# 16-bit input-output
00000 / 0 ;  # Put a value of zero into location 0
FFXX / 1234 ;  # Illegal addresses
FF00 / 1234 ;  # Put a value of 1234 into location FF00
FF00 / 11234 ;  # Illegal data
FF00 / 12XX ;  # Put a value of 12XX into location FF00
```

## **Initializing RAMs and ROMs**

The RAM and ROM models behave similarly, except that a ROM does not use read and write pins. The ROM model operates like it has a read pin tied to a logic 1 and a write pin tied to logic 0. If any X's appear on the RAM or ROM address bus, the device outputs all X's on the data bus.

When you use a ROM in a simulation, you must specify the ROM contents with the Modelfile property. Although, not required for a RAM (since you can write RAM contents during the simulation), you should initialize RAM contents in a similar manner. The Modelfile property gives the pathname to an ASCII file that contains initialization data. The format of the ROM/RAM initialization file is as follows:

- All data and addresses must be in hexadecimal, although X characters are allowed in order to specify unknown data values. Note that a single X value represents 4 bits of unknown value.
- Precede all comments with a pound sign (#).
- Specify the contents of a particular memory location using the form:

address / data; or low\_address-high\_address / data;

• Any letter can be in upper or lower case.

The previous page shows examples of ROM or RAM initialization files.

If you underspecify a data or address value, the simulator pads the value with zeros. Thus, for a 16-bit ROM, a data value of FF becomes 00FF. While you can legally overspecify data values with zeros or Xs, if overspecification causes a binary 1 to occur in a non-valid part of the data (for example, the eighth bit of a 7-bit data field) an error results. This is illustrated in the second example.



For more information on RAM and ROM models refer to the "Memory Devices (RAM, ROM)" section of the *Digital Simulators Reference Manual*.

# **How Models Are Registered**

Modeling Technique	<b>Registration Method</b>
Builtin Models	reg_model shell command
Schematic Models	Design Architect
QuickPart Table Models	qpt shell command
QuickPart Schematic Models	quickpart shell command
Memory Table Models	mtm shell command
BLMs	reg_model shell command
LM-Family Models	reg_model shell command
System-1076	Design Architect or hdl shell command
Technology Files	tc shell command

## **How Models Are Registered**

Mentor Graphics has provided automatic registration of models as a part of compiling a model or certain update processes. The table on the preceding page list the registration process for each of the modeling methods listed. The model types that are automatically registered during creation or compilation are:

- Schematic model. Design Architect registers this model as \$schematic.
- QuickPart Table model. The qpt shell command compiles the model and registers it with the interface.
- Memory Table model. The mtm shell command compiles the table and timing information and registers this model with the interface.
- System-1076. Design Architect allows you to compile source code, which registers with the component interface. The hdl shell command will also compile and register System-1076 models.

You can also change the model registration information using the reg\_model shell command. It allows you to change registration information, and to validate existing models.



For more information on how specific models are registered, refer to "Descriptor Registration" in the *Digital Modeling Guide*.

# **Verifying Models**



# **Verifying Models**

Model verification is a very important part of the simulation process. This activity provides you some assurance that your models behave consistently from one simulation to the next. If you don't verify models before a simulation run, you can't rely on the results to be accurate.

The first part of the process requires that the model creator (you, someone in your library management group, or the supplier of a component library) test and verify the proper function of the model. Once it functions properly, the setup conditions, stimulus, and results are saved with the component database.

The results must be in reproducible file form that can be compared to an equivalent file. This can be easily accomplished by writing list window information to a file. It is important that the List window be properly configured prior to simulation, and that this setup information also be saved.

Save a "README" or "commentfile" with the component to provide the model user sufficient information to test the model. This file may contain assumptions made during model creation, or other simulation specific information.

When you obtain a model to use in your simulation, you should verify it to make sure that it functions the way it was intended. This usually involves running the model in a test simulation using a supplied set of stimulus, and then comparing the results to known good results. In some cases, the job of verifying models before use in a simulation falls on the library management team or an external vendor. In other cases, the burden rests with the simulation engineer.

As a simulation engineer, you should also view the components you are simulating as models. You may be simulating an ASIC or module to be used in another design. As such, you should provide future users of your model a similar method of verifying that your electronic model works properly before it is used in another design.



For more information on model verification, refer to the "Verification of the Model" section in the *QuickPart Table Model Development Manual*.

# **Technology Files**

Notepad - \$HOME/qsim_n/LATCH/dff/technology.ts		
MODEL 'dff': TIMING =		
DECLARE		
#define CONTROL PRE, CLR		
BEGIN		
$\begin{array}{llllllllllllllllllllllllllllllllllll$	(L) (L)	;;
<pre>fMAX = 25 ON CLK; tS = 20.0 ON D(V) TO CLK (AH) WITH PRE,CLR (H); tH = 5.0 ON D(V) TO CLK (AH) WITH PRE,CLR (H); tW = 25.0 ON CLK (H); tW = 25.0 ON PRE,CLR (L); END ['dff': TIMING];</pre>		
	$\triangleright$	

# **Technology Files**

A Technology file contains timing and constraint information that is used together with a functional model. When QuickSim II is invoked in timing mode, technology files are examined and compiled to be used during the simulation run. The basic structure of a timing file is shown on the previous page, together with several types of typical statements.

In most cases, the simulation engineer does not manipulate technology file. This is the job of the model developer. Components in libraries that are supplied by Mentor Graphics and other library vendors will normally contain Technology files that are valid in QuickSim II and other analysis tools. When models are developed the customer site, a model development group usually develops the Technology file for the engineering group.

The statements in a Technology file that affect your simulation are:

- **tP or Delay**. It is used to create pin-to-pin delays in your model. These delays may be combined with Rise and Fall property delays on the instance.
- **tS or Setup**. This statement is enabled with constraint checking and warns you if signals are not stable prior to a clocking signal.
- **tH or Hold**. This constraint statement warns you if a signal changes during a period of time after a clocking signal.
- **tW or Width**. This constraint statement warns you if the pulse of a signal is shorter than specified.
- Fmin and Fmax. These statements warn you when a signal's instantaneous frequency goes below (fmin) or above (fmax) a specified frequency



For more information on the structure of a Technology file, refer to the *Technology File Development Manual*.

# Lab Preview

- Examine components that contain
  - Builtin models
  - Sheet-based schematic models
  - QuickPart Schematic models
  - QuickPart Table models
- Examine the navigation icons
- Use Component Interface Browser to examine a component interface
- Examine the contents of a QuickPart Table model
- Create and compile a Technology file

### Lab Preview

In the lab exercise for this module, you will:

- Examine component libraries that contain the following types of component models:
  - Builtin models
  - Sheet-based models
  - QuickPart Schematic models
  - QuickPart Table models
- Examine the Navigation icons that represent design objects. Each type of object has a unique icon that identifies it. You will examine these objects.
- Use the Component Interface Browser (CIB) to examine a component interface.
- Examine the contents of the QuickPart Table model for the "dff" component.
- Create and compile a technology file, and register it with the component interface. You will also validate existing interfaces.

### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

### **Procedure 1: Examining Models**

This lab exercise will help you become acquainted with the V8 component interface and modeling structure.

- 1. Log into your workstation using your account and password.
- 2. If the Design Manager is not automatically invoked, invoke it in a shell as follows:

shell> \$MGC\_HOME/bin/dmgr

3. Examine the contents of the "pullup" component using the Design Manager by doing the following:

Open the iconic navigator window, if not already open. Click on the "Go

To" button at the bottom of the iconic navigator window.

Enter the following path in the destination field.



	/trai	ning/qsir	n_n/lib/pull	up <u>-</u>	
C	part ommentfile	pullup	Schematic		
Selected: 0					

You should see the objects, including "schematic", as shown. What type of functional model do you suppose this component uses, by default?

4. Examine the contents of the "dff" component as follows:

Using the Design Manager navigator window, navigate to the following location:

#### \$HOME/training/qsim\_n/lib/dff



You should see a TECH icon named "technology" which is the compiled technology file for the QuickPart. You will also see the source for the technology file (technology.ts), the QuickPart Table model ASCII file (model.table), and the model verification file (sim.do).

What type of functional simulation model do you suppose this component uses, by default?

5. Use CIB to examine the dff component that is used in later simulations by entering the following in a new shell:

```
$HOME/training/qsim_n
shell> $MGC_HOME/bin/cib
CIB > open $HOME/training/qsim_n/lib/dff
dff::dff > view
```

Examine the listing. It is provided on the next page for reference.

	/user/proj	ject/designs/	ibrary/dff	
COMPONENT di	ff	DEF	AULT INTERFACE IS: df	f
INTERFACE:	dff			
PINS:	Compiled	User		
Id #	Pin Name	Pin Name	Properties	
1	CLK	CLK	(pintype, IN)	
			(pin, CLK)	
2	D	D	(pintype, ENA)	
			(pin, D)	
3	QB	QB	(pintype, OUT)	
			(pin, QB)	
4	Q	Q	(pintype, OUT)	
			(pin, Q)	
5	CLR	CLR	(pintype, ENA)	
			(pin, CLR)	
б	PRE	PRE	(pintype, ENA)	
			(pin, PRE)	
BODY PRO	PERTIES:		(model, \$gen_qpt)	
			(qrise, 0)	
			(qtall, 0)	
			(qbrise, 0)	
			(qbiall, 0)	
			(qp_prim, TRUE)	
INTERFAC.	E MODEL EN	TRIES:		
Model En	try Type	1 1	Model Inio	
U	mgc_s	YMDOT	Path:/dii/dii	
Labels	· derault	_sym'		
Status	• valid to	r interiac	e, valid for prope	erty
L Tabala	uugc_s	tial lacha	Patili/dil/Schemat	.10
	· sschema · valid fo	r interfac	Malid for prope	wt.r
SLALUS	• Vallu lo		Dath: /dff/tachnal	erty
z Labels	: 'def tec	h' 'Saen a	pation / arr/technor pt'	.ogy
Status	: Valid fo	r interfac	e; Valid for prope	ertv
3	OPT		Path:/dff/qpfile	- 1
Labels	: '\$qen ap	t'		
Status	: Valid fo	r interfac	e; Valid for prope	erty

Note that four model entries exist. Model entry 0 is for the symbol. Model entry 1 is for the schematic. Model entry 2 references the "technology" and is of type Technology. Model entry 3 references the compiled QuickPart Table model. Also notice that under "BODY PROPERTIES" the model property is set to "\$gen\_qpt" which is the label for the QPT model and the technology file.

6. Examine the contents of the "rip" component using the Design Manager.



#### Path: **\$HOME/training/qsim\_n/lib/rip**

It has eleven symbols that can be used for this component. Which one is the default symbol? \_\_\_\_\_\_ (You'll verify this in the next step.)

What functional model is used for simulation?

- 7. From the Design Manager, open QuickSim II on the LATCH component as described in the following steps:
  - a. In the Design Manager Tools window, double-click on the QuickSim II
    - tool icon The QuickSim II dialog box appears.
  - b. Fill out the dialog box as shown:

QuickSim II		
Design	\$HOME/training/qsim_n/LATCH Navigator	
Symbol	Interface	
Timing mode     Previous     Unit     Delay     Constraint       Detail of 'Delay' timing mode     Hidden     Visible		
Simulator resolution 0.1 ns		
OK Reset Cancel		

c. **OK** the dialog box.

QuickSim II invokes on the LATCH design using the design viewpoint that you created in the DVE lab.

- 8. Now use the QuickSim II session window to examine the "rip" component interface using the Component Window, by doing the following steps:
  - a. Choose: MGC > Design Management > Open Component Window

The Open Component Window dialog box appears.

b. Select the "lib" component in the navigator window and **OK** the dialog box.

The Component Window appears on the left side of the screen with only the "lib" container displayed in the Component Information window.

c. Double-click on the "lib" container to reveal library objects beneath it.

You should see the tree of components provided with the training data. One of the components is the bus ripper "rip"

d. Double-click on the "rip" component to reveal its structure.

The interfaces are listed first, then the component symbols, as shown.



e. Click on the 1X2 (Default) interface. The Registered Model Info window shows that there is only a symbol listed for this object. This is a primitive

object and does not require a functional model. No simulation model exists, since the rip component is for connection only.



- f. Click on several other "rip" interfaces and examine them.
- g. Double-click on the "dff" container in the Component Information window. Information for the "rip" component is removed and the dff component information is expanded.
- h. Click on the "dff (Default)" interface. The interface information is presented in the Registered Model Info window.
- i. Finally, double-click on the Timing Model's symbol (for Technology). This shows the registered labels for this model. Double-click on several of the other models to examine their labels.



- 9. Next, you will examine "dff" QuickPart Table using Notepad.
  - a. In the QuickSim II session, set the default font for the Notepad editor as follows:

#### MGC > Notepad > Set Default Font > Monospace > 14 pt bold

This presets the Notepad window font to an equal-spaced font so that columns in the table properly align.

b. In the Component Information window, double-click on the "model.table" object contained within "dff."

The Notepad window appears with the model.table file displayed. The object named "model.table" is the state table ASCII file for the QuickPart Table model. It is copied below for convenience. Line numbers (italic text) have been inserted for description purposes.

```
Notepad - $HOME/training/lib/dff/model.table (R)
                                                              01 MODEL 'dff': TABLE =
   02
   03 INPUT CLR, PRE, D;
   04 EDGE_SENSE INPUT CLK;
   05
   06 OUTPUT Q, QB;
   07
   08 STATE_TABLE CLR, PRE, CLK, D, Q, QB :: Q, QB;
   09
   10
                     Ο,
                          1,
                               [??],?,
                                        ?, ?
                                              :: 0, 1;
                                        ?,
                                           ?
                                              :: 1, 0;
   11
                          Ο,
                               [??],?,
                     1,
                     Ο,
                          Ο,
                                              :: 1, 1;
                               [??],?,
                                       ?, ?
   12
                     1,
                               [01],?, ?, ?
   13
                          1,
                                              ::(D),!(D);
                     1,
                          1,
                               [?0],?, 0, ?
                                              :: N, N;
   14
                                              :: N,
                               [?0],?, ?,
                                           0
   15
                     1,
                          1,
                                                     N;
                               [1?],?, 0, ?
                                              :: N,
   16
                     1,
                          1,
                                                     N;
                     1,
                               [1?],?, ?,
                                           0
                                              :: N, N;
   17
                          1,
                          1,
                               [?0],?, !0,!0 :: X, X;
   18
                     1,
   19
                     1,
                          1,
                               [1?],?, !0,!0
                                              :: X, X;
   20
                     Х,
                          ?,
                               [??],?, ?, ?
                                              :: X, X;
                                              :: X, X;
   21
                     ?,
                          Х,
                               [??],?, ?, ?
   22
   23 END ['dff': TABLE];
```

Line #04 declares CLK as type "edge\_sense".

Note that the column for CLK contains two states that define the edge. For example, line #13 occurs when the CLK signal transitions from 0 to 1, placing the D input on the output Q and the opposite of D (! is the NOT operator) on QB.

What line number describes the behavior of the dff when the CLR signal is unknown? \_\_\_\_\_

What are the outputs under this condition?

c. Close the Notepad window: (Notepad Window Menu) > Close

Do NOT close the Component Window. You will use it in the next procedure.

### **Procedure 2: Verifying Models**

In this procedure you will use the techniques you learned on page 4-28 to determine if the LATCH model (dff) performs correctly according to the sim.list file.

1. Double-click on the "sim.do" object beneath the "dff" component in the Component Information window.



The Notepad window appears containing the sim.do file. Examine its contents, noting the force and run functions.

```
Notepad - $HOME/training/qsim_n/lib/dff/sim.do
...<information omitted>...
     $add_lists(@change, @binary, void, 'CLR', 'PRE',
   'CLK', 'D', 'Q', 'QB');
     $run(50);
         //// NORMAL FUNCTIONALITY
     FORCe 'CLR' '1';
     FORCe 'PRE' '1';
     FORCe 'CLK' '0';
     $run(50);
     FORCe 'D' 'Or';
     $run(50);
     FORCe 'CLK' '1';
     $run(50);
     FORCe 'D' 'Xr';
```

2. Close the Notepad window and the Component Window.

3. Execute the *sim.do* AMPLE script by entering the following on the QuickSim II popup command line (which appears when you start typing):



This script creates a schematic view window and a List window, and then issues stimulus to the LATCH circuit (the dff component model) at 50 nsec. time increments. The run stops at time 4700.

- 4. Write the List window results to a file as follows:
  - a. Activate the List window by clicking the Select mouse button in it.
  - b. Choose: (Menu bar) > File > Write

The Write Report dialog box appears. This window allows you to write the information displayed in the window to an ASCII file.

c. Enter the following pathname and **OK** the dialog box:

Write Report						
Pathnameqsim_n/LATCH/sim.tes	Navigator					
Default Replace Append	Default Replace Append					
Format	Highlights					
<ul> <li>To look like window</li> <li>With page settings</li> <li>Without truncating data</li> </ul>	Show Hide					
Begin time End Time						
Local header						
Include session header						
OK Reset Cancel Help						

If you do not specify a start or stop time, all information will be written to the file.

- 5. Using the Notepad editor, examine this *sim.test1* file as follows:
  - a. Choose: MGC > Notepad > Open > Read-only
  - b. When the "Select file to view" navigator box appears, navigate beneath the LATCH object. (You may need to change the "Filter" to see the LATCH object) Then select the "sim.test1" object.
  - c. **OK** the navigator box.
  - d. Now examine the contents of the Notepad window (shown below).

□ Notepad	- \$HOM	IE/trai	ning/qsim_n/LATCH/sim.test	
0.0 Xr X:	Xr X:	r Xr	Xr	$\square$
0.1 Xr X:	Xr X:	rХ	Х	
50.0 1 1	0 X:	rХ	Х	
100.0 1 1	0 0:	rХ	Х	
150.0 1 1	1 0:	rХ	Х	
150.1 1 1	1 0:	r 0	1	
(omitte	1)			
4400.0 1 0	X X	1	0	
4450.0 1 0	0 0	1	0	
4500.0 1 0	0 1	1	0	
4550.0 1 0	X 1	1	0	
4600.0 1 0	0 X	1	0	
4650.0 1 0	1 X	1	0	
Time(ns)^CLR	^CLK	^Q	^QB	
<u>^</u>	PRE ^	D		
				$\geq$

It is a text copy of the information in the List window, including the signal names at the bottom of the file.

e. Close this Notepad window using the Window Menu button.

- 6. Exit QuickSim II without saving the results as follows:
  - a. Choose: (QuickSim II Window Menu) > Close
  - b. Click on "Without saving" and **OK** the dialog box.

Exit QuickSim			
After Saving Without saving			
OK Reset Cancel			

QuickSim II exits, and the shell transcript remains.

- c. Remove the shell transcript pad that remains.
- 7. Compare the two List window output files to validate results by using your operating system command that compares the contents of two ASCII files located at:

#### \$HOME/training/qsim\_n/LATCH/sim.test1 \$HOME/training/qsim\_n/lib/dff/sim.list

For example, if you are using a UNIX based workstation, you can issue the following command if you are in the LATCH directory:

SHOME/training/qsim_n/LATCH					
<pre>shell&gt; diff sim.test1/lib/dff/sim.list shell&gt; (no message indicates compare was successful)</pre>					

If the compare is successful (prompt returns without transcripting any messages), you have validated the model. If it is unsuccessful (miscompares are transcripted), it may indicate model *or* simulation setup problems.

### **Procedure 3: Changing a Technology File**

In this subsection, you will create a Technology file, and then compile and register it with the LATCH (dff) component.

1. From within the Design Manager, change your working directory to that of the LATCH circuit (directory), as follows:

#### Choose: (Menu Bar) > MGC > Location Map > Set Working Directory

The prompt bar appears containing the current working directory path:

SET WO D Directory \$HOME/training/qsim\_n OK Cancel

Enter the following pathname as shown and **OK** the prompt bar:

SET WO D Directory \$HOME/training/qsim\_n/LATCH OK Cancel

Next you will edit the Technology file for the dff component so that it contains typical timing values similar to the 74ls74 device.

- 2. Open the Notepad on the dff "technology.ts" file as follows:
  - a. Using the Design Manager Navigator window, locate "technology.ts" beneath the *\$HOME/training/qsim\_n/LATCH/dff* directory.

Notice that this dff component has a different path than the one previously viewed. This dff component is beneath "LATCH."

b. Select this "technology.ts" object

#### c. Choose: (popup) > Open > Default Editor

The Notepad opens in edit mode on the technology source file for dff.

Note that the timing values are set to zero (the four statements in the DECLARE section). This is typical for Gen\_lib library components like "dff".

3. Next you will create a new technology file that look like the one in this Notepad window by doing the following steps:

Notepad - \$HOME/training/qsim_n/LATCH/dff/technology.ts		
MODEL 'dff': TIMING =		$\square$
DECLARE		
#define CONTROL PRE, CLR		
BEGIN		
<pre>tP = 6.5, 13, 25 ON CLR (AL) TO QB (AH); tP = 12.5, 25, 40 ON CLR (AL) TO Q (AL); tP = 6.5, 13, 25 ON PRE (AL) TO Q (AL); tP = 12.5, 25, 40 ON PRE (AL) TO QB (AL); tP = 12.5, 25, 40 ON CLR (AH) TO QB (AL) WITH PRE (L) tP = 12.5, 25, 40 ON PRE (AH) TO Q (AL) WITH CLR (L) tP = 6.5, 13, 25 ON CLK (AH) TO Q,QB (AH); tP = 12.5, 25, 40 ON CLK (AH) TO Q,QB (AL);</pre>	;;;	
<pre>fMAX = 25 ON CLK; tS = 20.0 ON D(V) TO CLK (AH) WITH PRE,CLR (H); tH = 5.0 ON D(V) TO CLK (AH) WITH PRE,CLR (H); tW = 25.0 ON CLK (H); tW = 25.0 ON PRE,CLR (L); END ['dff': TIMING];</pre>		

- a. In a new Notepad window, open the "technology.edits" source file readonly (located in the same directory as the "technology.ts" object).
- b. Select all of the contents of the technology.edits file. Note that header and end statements are missing.
- c. Copy the selected contents to the session paste buffer using the following menu item:

#### (popup) > Copy to Clipboard > Session

- d. Close the Notepad window that is opened on "technology.edits"
- e. Select and delete all but the header and end line from "technology.ts"

f. Paste the session buffer into the "technology.ts" file and perform minor edits so that technology.ts resembles the previous listing.

#### (popup) > Paste > Session

Notice that both square brackets ([]) and parentheses are used in the file. The Technology Compiler expects square brackets to only in specific areas of the Technology file, specifically, the END statement.

The delay times that are used in the dff Technology file were determined using the 74ls74a model. The setup (TS) and hold (TH) constraint values, and the minimum signal pulse width (TW) were determined after an analysis of the circuit requirements.

g. Now save the file as follows:

#### (Notepad Window menu) > Close

- h. When the Save Changes question box appears, choose YES.
- 4. Compile the "dff" Technology file using the TC command as follows:
  - a. Enter the following shell command in a new shell:

<b>CUONE</b>	/troining	/acim	
	/11/2011010		
	/ LI GII III I G		
			—

The compile switches do the following:

- -r Replace -- changes or removes labels, overwrites design objects
- -lst Creates a listing file from the compiler (shows compile details)
- -va Revalidates the interface after the part interface is changed

Notice in the transcript that the label \$gen\_qpt was removed from the technology model. The technology model needs this label to properly attach the compiled technology (timing) information to the functional QPT model. The next step recompiles the model to add the \$gen\_qpt label to the technology model entry.

5. Recompile so that the label \$gen\_qpt is valid by issuing the following:

```
$HOME/training/qsim_n/LATCH
shell> $MGC_HOME/bin/tc $HOME/training/qsim_n/LATCH/dff
-r -lst -va -label 'def_tech' -label '$gen_qpt'
```

The Technology Compiler generates the compiled files required by QuickSim II, and reports on its progress while executing.

You have now made a primitive of the dff component by compiling and registering technology.ts file, and validating the interface. You now have real timing values for this component.

6. Exit the Design Manager.



LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 4 Solutions" on page A-7.

Then continue on to the next module.

## **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 4 Solutions" on page A-7.

- 1. Which symbol property provides a path to RAM or ROM initialization files? (Circle one) Model Modelcode Modelfile Comp
- 2. Define component registration: \_\_\_\_\_
- 3. The value of the \_\_\_\_\_ property is matched against existing component interface labels to determine model selection.
- 4. Overloading refers to the process of creating your own models for the \_\_\_\_\_\_ models.
- 5. Which of the following are supported in QuickSim II? (circle all that apply).
  - a. QuickPart Table models
  - b. QuickPart use of primitive switches
  - c. QuickPart hierarchical models
- 6. True or False: Behavioral Language Models (BLMs) can use technology files for timing and technology information.
- 7. True or False: VHDL models and other modeling techniques cannot be mixed in the same schematic design.

# **Module Summary**

This module describes V8 modeling using the following categories:

- **Component interface concept.** The component interface is contained in the "part" object. Models are registered with this interface. The model property is assigned a registered value, called a label, which maps to one of the available models. Changing a model is as simple as back annotating a new model property.
- **Component Interface Browser (CIB).** This shell-based tool allows you to examine and edit component interfaces. The text is formatted and displayed in the shell pad.

• Modeling Methods:

- Builtin models. The functionality is builtin to QuickSim II.
- QuickPart Schematic models. Schematics and timing are compiled.
- **QuickPart Table models**. Most Mentor Graphics libraries use memory efficient QuickPart Table models.
- Memory Table Models.
- **VHDL Support.** The System-1076 implementation of VHDL language is used by Mentor Graphics for creating numerous models with a high level of abstraction. Full VHDL debugging support is provided with QuickSim II.
- **Hardware modeling**. The LM-Family of modelers use real devices to create simulation models. Functionality and timing is provided.
- **Technology files.** The technology file contains timing information and includes properties related to design technologies. Capacitance, temperature, and other characteristics can be modeled. Technology files can be used by most modeling methods including BLMs, QuickParts, builtin models, and sheet-based models.
# Module 5 Setup and Expressions

Overview	5-2
Lesson	5-3
Simulation Setup	5-4
Setting Up the Session (review)	5-6
Setting Up the SimView Interface	5-8
Setting Up the Environment	5-10
Setting Up the Kernel	5-12
Reporting Setup Conditions	5-14
Saving Setup Conditions	5-16
Restoring Setup Conditions	5-18
QuickSim II Startup Files	5-20
Keeping Circuit Activity	5-22
Simulation Timing Modes	5-24
Scaling Delays	5-26
Simulation Expressions	5-28
Expression Examples	5-30
Lab Preview	5-34
Lab Exercise	5-36
Procedure 1: Setting Up QuickSim II	5-36
Procedure 2: Restoring the SimView Setup	5-44
Procedure 3: Creating and Using Expressions	5-46
Test Your Knowledge	5-52
Module Summary	5-53

### **Overview**



## Lesson

Upon completion of this module, you will be able to:

- Set up your QuickSim II and SimView session environments, including window placement and visibility, mouse type and double-click speed.
- Set up the QuickSim II and SimView environments to include default user units and radix, force type, exit queries, and scale factor.
- Create standard report windows containing setup conditions, object info, timing info, design changes, and simulation messages.
- Set up the kernel to adjust the simulation kernel mode of operation, and to use the kernel setup operation to examine the current setup.
- Save and the restore setup conditions for QuickSim II.
- Describe how startup files are used within QuickSim II invocation, and create a startup file in the proper location to automatically be used during invocation.
- Add Keep signals to the simulation so that the data for those signals is recorded in the "results" waveform database.
- Scale the delays that are provided to QuickSim II so that you can adjust typical timing to be used a Min or Max timing values.
- Create simulation expressions that you can use within breakpoints, and you can trace or list. Several expression examples are provided.



You should allow approximately 1 hour and 15 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

## **Simulation Setup**

Three types of setups:

- Session
  - Common to all Mentor Graphics applications
- User interface (SimView/UI)
  - Common to all Mentor Graphics analysis applications
- Kernel (A simulator connected to SimView/UI)
  - Each simulator kernel is unique

## **Simulation Setup**

There are three areas of the SimView/UI-based simulation environment that you might be interested in setting up:

- The session. Common to all Mentor Graphics applications
- **The SimView user interface**. Common to all Mentor Graphics *analysis* applications
- **The simulator kernel**. The setup for each simulator is unique. Refer to your simulator's documentation for setup information.

## Setting Up the Session (review)

Click on: [Setup palette]



- 1	
	Show Transcript
	Setup Session
	Select Windows
	<b>Cleanup Windows</b>
	<b>Cleanup Windows</b>
	All Windows
	Minimize
	Restore
	Close



## **Setting Up the Session**

Setting up the session window for simulation is the same as for most other Mentor Graphics applications.

To set up the simulation session window, you can use the **MGC** > **Setup** > **Session** pulldown menu item, or the **[Setup] Setup Session** palette button. The following list contains some of the choices you can make concerning the session setup:

- The type of graphic input device used, such as a mouse or a puck. The default is a mouse.
- The speed that a double-click is detected. The default is the fastest available speed.
- Aspects that control how much of the session window is available for application windows. The default is to show all of the window areas. You can hide these areas to gain more room for windows that the simulator generates. As you already discovered, you can hide palettes and softkeys using the popup menus.
- The window layout, such as stacking or tiling. The default window layout is stacking (consecutive windows are on top of each other and slightly offset).



For more information about setting up the session window, refer to the *Common User Interface Manual*.

## Setting Up the SimView Interface

### • From the Setup menu:



 You can place setup commands in a startup file or macro

## Setting Up the SimView Interface

A Mentor Graphics simulator is partitioned into two fundamental parts: the user interface (SimView/UI) and the kernel. To manually set up either part, you can use the **Setup** pulldown menu.

If you don't want to manually do the setup each time, you can create startup files or dofiles that automatically set up your session for you.

The first choice in this menu is for setting up the kernel, and is described in the kernel's documentation. The remaining menu choices apply setup conditions to the user interface. The following list contains a brief description of the choices:

- **Keeps**. Sets the default conditions for Adding and Deleting Keeps. Although a simulator must be present for this function to operate, Keeps are included here because they are a part of the SimView/UI.
- **Environment**. Used to set the user time scale, the keeps defaults, exit queries, the default force type, and the default radix.
- Set Default. Sets the default waveform database, force target, and other objects.
- **Stimulus section**. Various stimulus-related actions are presented in this section, including various forces, waveform creation, and waveform database connection and disconnect.
- Set|Clear Selection Filter. Set or clear specific selection criteria.
- Window Attributes. Hides window scroll bars and titles, configures the Trace, List, and other windows.
- Open New Window. Create empty Trace, List, or Monitor window.
- For specific information about setting the conditions for the user interface, refer to the "Operating Procedures" section of the *SimView Common Simulation User's Manual*. Using startup files and dofiles is described in the *Common User Interface Manual*.

## **Setting Up the Environment**

### Choose: (Menu bar) > Setup > Environment

User Scale Type	e time	Scale 1e-09
Write Report Page width 80 Page length 66 Session Header		
Exit querys Designs Setup WDBs Timing	Force type Fixed Wired Charge Old	Radix Hex Cotal Binary Decimal Float Signed

## **Setting Up the Environment**

The SimView environment determines the user perspective for the user interface. This is the mechanism you use to set up the following default values: user units, radix, keep list, force type, as well as others. To change any of the environment default setups, choose:

### (Menu bar) > Setup > Environment

The dialog box appears as shown. Contained within this dialog box are the following setups:

• User Scale. This specifies how time entries are interpreted in commands, prompt bars, and dialog boxes. The initial default units are nanoseconds. Changing this setup does not affect the property delays or technology file times, which are always in nanoseconds.

Note: The simulator time step, which affects kernel resolution, is set on invocation only, and is independent of the user scale.

- Write Report. Defines printer setup information.
- **Exit queries.** QuickSim II determines what you have changed during the simulation and can prompt you to save changes when you exit. You enable or disable the prompts with this item.
- Force type. Determines the default force type when the Force command is issued. The initial default is "Charge".
- **Radix.** Determines how signal values are interpreted. The initial default radix is "Hex".

You can explicitly specify the radix of each entry as follows:Binary - "0b" prefix. Example: 0b101 (= 5 decimal)Octal - "00" prefix. Example: 0o101 (= 65 decimal)Decimal - "0d" prefix. Example: 0d101 (=101 decimal)Hex - "0x" prefix. Example: 0x101 (= 257 decimal)

## **Setting Up the Kernel**

• (Menu bar) > Setup > Kernel (> Analysis)

Setup Analysis		
Timing modeCurrentUnitDelayConstraintDetails of 'Current' timing modeHiddenVisible		
Timing mode =unitChangeDelay Scale1Constraint modeOffState onlyMessagesSpike modelX immediateSuppress	_Override _Override _Override	
Hazard checkOverrideSpike warnings toContention checkOverrideSuppressModel messagesOverrideXToggle checkOverride	display: Override	
OK Reset Cancel		

If you click on Change	Unit Delays			
	Full Delays	Min	Тур	Max
	Linear Delays	Min	Тур	Max

• You can change conditions on instances

(Menu bar) > Setup > Kernel > Change > (item)

## **Setting Up the Kernel**

At the kernel (or simulation) level you can set up the timing mode to be used on the entire design. To setup the kernel for the entire design choose the **Setup** > **Kernel** > **Analysis** menu item. This menu item presents a dialog box with the following choices:

- **Timing mode**. Min, Typ and Max use values from the triplicate set of timing numbers. Unit (the invocation default) uses a delay of one timestep for each gate, while timing is ignored. Linear Delays are linear timing values, which use straight-line approximations for the min, typ, and max values.
- **Constraint mode.** Setup, hold, stability, pulse-width, and minimum and maximum frequency checking.
- Spike model and Spike check. X-immediate or suppress models.
- **Hazard check**. A hazard condition exists when an output or IO pin changes to two or more different states during the same timestep.
- Contention check. Two or more net drivers on the same net.
- **Model messages**. You can design a QuickPart Table to generate messages if certain conditions occur during simulation. This can aid in debugging your design. Note that displaying QuickPart Table messages may degrade the simulator performance.
- **Toggle check**. To measure how many times signals toggle between 1 and 0, you can gather toggle statistics. Signal toggling statistics are useful in estimating how effective your functional verification stimulus will be for detecting faults in fault analysis applications.

In addition to setting up the entire design for a simulation, you can specify setup conditions for individual instances. This is particularly useful when your analysis is focused at a lower level in the design. To setup one or more individual instances, you select the desired instance or instances and change the timing mode, using the **Setup > Kernel > Change >** [item] menu item.

# **Reporting Setup Conditions**

Poport

- Support:
  - Cross-window highlighting
  - Source viewing
- Report\_name (two parts)
  - Window\_name
  - Optional path

report	
Setup       ▶         Waveform DBs       ₩aveforms         Waveforms       SimView Assertions ▷         Design Changes       ▷         Design Changes       ▷         Design Messages       ▷         Toggle       Timing         Timing Cache       Model Statistics         Interfaces:       ¬         References       ¬	StimulusBreakpointsActionpointsActionpointsKeepsProbesWDB FiltersBus DefinitionsFormattersExpressionsSetup GroupsCurve Styles
Object	

### **Example of a Report Window**

□ Keeps				
Object name	Туре	Full	Windowed	A
/PRE	add	т	Т	
/CLR	add	т	F	
/D	implicit	F	F	
/CLK	implicit	F	F	
/QB	implicit	F	F	
/Q	implicit	F	F	
				Δ

## **Reporting Setup Conditions**

Report windows are a convenient way of organizing information into useful categories. Report windows are named by the type of information they contain, to make window management easy. If you create a second report window of the same type before the first window is removed, a window sequence number is appended to the report window name, such as "Objects#2".

In some cases, the report\_name is followed by a path to the object that the report reveals (not shown). This pathname is useful if you want to learn where these objects are stored, or you have multiple objects, and you want to verify that you are viewing the correct object.

To create a report window:

#### (Menu Bar) > Report > report\_name

The reports that are available are shown in the report menu.

Report windows support cross-window highlighting. For example, the stimulus of a waveform database is being displayed in the Trace window. If you open a report window on a waveform database stimulus object, and then select the stimulus name in the report, the trace(s) will highlight in the Trace window.

## **Saving Setup Conditions**

Save Set	up
Viewpoint	
Pathname quicksim_setup	Navigator
Replace	
Query when Waveform DBs	s have edits pending
Save All System Setup Gro Specify the System and/or User defined setup groups to save User defined setup groups	ups System setup groups actionpoints assertions breakpoints buses chart_windows comment_flags context cycles design_messag expressions hdl_setup hier_modes keeps list_windows model_load monitor_flags monitor_flags monitor_flags monitor_flags monitor_flags monitor_flags monitor_window probes qs_parameters run_setup session_attribut simview_attribu source_views synonyms trace_windows warn_start wdb_filters
OK Reset Car	ncel Help

• Groups are saved in setup data, default=all

## **Saving Setup Conditions**

You can save your setup conditions in a design object in order to reuse a stable setup environment that you have established. Each time you invoke or reset the simulator, you can restore the setup.

To save a setup, use the **File** > **Save** > **Setup** menu item. This menu item presents a dialog box with the following choices:

- Viewpoint. A button that enables saving the setup object in the design viewpoint directory. This feature lets you keep specific setups with specific design viewpoints. If you click on this button, you must also provide a leafname for the setup object.
- **Pathname**. An input field that specifies the pathname for the setup object. If the Viewpoint button is enabled, the Pathname input field is not displayed, because it is unnecessary.
- **Navigator**. A button that displays a file system navigator that you can use to inspect your file system. If the Viewpoint button is enabled, the Navigator button is not displayed, because it is unnecessary.
- **Replace**. A button that allows an existing setup to be overwritten with the current setup. If you do not select this button and an existing setup by the same name already exists, a message is displayed notifying you that the save was not performed.
- Query when Waveform DBs have edits pending. A button that allows you to turn off query messages regarding waveform databases that have not been saved since the last edit operation. If you turn off querying and an edited waveform is inadvertently not saved, the edits will be lost.
- Save All System Setup Groups. A button that allows you to either save all the setup groups or to save selected groups from a list provided. If you don't specify any groups, all groups are saved.

## **Restoring Setup Conditions**

Restore Setup				
Viewpoint				
Pathname quicksim_setup	Navigator			
Restore setup without confirmation				
<b>Do NOT restore WDBs</b> (Supercedes manual group selection.)				
Restore All System Setup Gr Specify any combination of System, User defined, or Other setup groups to restore. User defined setup groups	oups         System setup groups         actionpoints         assertions         breakpoints         buses         chart_windows         comment_flags         context         cycles         design_message_         expressions			
Other setup groups Defined in saved object Group	hdl_setup hier_modes keeps list_windows model_load			
OK Reset Ca	ancel Help			

- A dialog box appears containing:
  - Viewpoint
  - Pathname & Navigator
  - Group selection (Default, all groups restored)

## **Restoring Setup Conditions**

You restore saved setup conditions whenever you want to return to a setup environment. The setup saved contains both window and kernel setup conditions, such as timing mode, delay scale, checking modes, and spike and contention models. Some situations in which you may want to restore setup conditions are: each time you invoke, whenever you reset the simulator to time zero, or anytime during a session that you want to return to a known setup condition.

To restore saved setup conditions, use the **File > Restore > Setup** menu item. This menu item presents a dialog box with the following choices:

- Viewpoint. A button that enables restoring the setup object from the design viewpoint container. This feature supports keeping specific setups with specific design viewpoints. If you click on this button, the simulator lists the contents of the design viewpoint container, from which you can choose the setup object.
- **Pathname**. An input field that specifies the pathname for the setup object. If the Viewpoint button is enabled, the Pathname input field is not displayed.
- **Navigator**. A button that displays a file system navigator that you can use to inspect your file system. If the Viewpoint button is enabled, the Navigator button is not displayed because it is unnecessary.
- **Groups**. An input field that specifies the setup groups you want restored from the setup. If you don't specify any groups, all groups are restored as a default.

Note that Restore Setup is not the same as Restore State. The Restore State command restores the signal states to each net and pin in the design. This is so you can continue simulating where you left off. Saving and restoring simulation states is discussed in "Saving and Restoring Simulation States" on page 8-20.



For more information on saving and restoring the setup conditions, refer to "Entering and Exiting a Simulation" in the *SimView Common Simulation User's Manual*.

## **QuickSim II Startup Files**

Startup file -- AMPLE program that executes specific actions automatically when you invoke application

1. Site-specific startup files:

\$MGC\_HOME/shared/etc/cust/startup /appl\_name.startup

- 2. Workstation-specific startup files: \$MGC\_HOME/etc/cust/startup/appl\_name.startup
- 3. User-specific startup files:

(\$HOME)/mgc/startup/appl\_name.startup

4. Design-specific startup files:

design\_path/viewpoint\_name/appl\_name.startup

## **QuickSim II Startup Files**

A startup file is an AMPLE program that allows you to execute specified actions automatically when you invoke an application. All startup files are written in AMPLE and can contain complex code.

There are four locations for application startup files, that applications execute in the following order:

1. Site-specific startup files: These files are customized to invoke applications for your workplace. The default path is:

*\$MGC\_HOME/shared/etc/cust/startup/*application\_name.*startup* 

2. Workstation-specific startup files: These files are customized to invoke applications for your type of workstation. The default path is:

*\$MGC\_HOME/etc/cust/startup/*application\_name.*startup* 

3. User-specific startup files: These files are customized to suit your personal working environment and are the files you are most likely to modify or have modified. See your system manager for assistance in modifying this file if you are not familiar with AMPLE. The default path is:

(\$HOME)/mgc/startup/application\_name.startup

4. Design-specific startup files: You create these files to setup an application to the specific needs of a design.

design\_path/viewpoint\_name/application\_name.startup

Remember, any of these files that exist when you invoke the application will be executed in the above order. Therefore, definitions in the design-specific startup file could override definitions in site- and workstation-specific startup files.



Refer to *Customizing the Common User Interface* for a detailed discussion of startup files and your application manuals for example startup files for your application.

# **Keeping Circuit Activity**

- Applies to simulators connected to SimView/UI
- All circuit activity in waveform database.
- *Keep list --* signal names stored in "results" waveform database.
- To add keep signals: [Setup palette]

Add Keeps		
On Selected Objects Named objects		
Window (Keep 0.0ns of signal history.)		
Full (Keep full debug information.)		
Setup Keeps		
OK Reset Cancel Help		

- Delete > Keeps menu item removes kept signals
- Keeping instances -- keeps all nets below instance
- A moving window of History
  - Add Keep command with -Window switch.
  - Maintained within kernel (instead of the results waveform database)
  - Is NOT saved when you exit QuickSim II

## **Keeping Circuit Activity**

The concept of keeping circuit activity only applies to simulation kernels operating with SimView/UI such as, QuickSim II. The simulation kernel, by default, maintains only the current and scheduled event information about the design circuit; all history information is discarded. "Keeps" create a *keep list* and instruct the simulation kernel to save the history of events for specific design objects. The keep list contains the name of each design object whose activity you want to store in the results waveform database. There are two ways to add objects to the keep list:

- Using the Add Keeps command (with the -Full switch, the waveform database contains pin states, net states, and force states for the specified objects). This is referenced as an *explicit* keep.
- By displaying or accessing the object (tracing, listing, monitoring, expressions, breakpoints). This is referenced as an *implicit* keep.

The Delete Keeps command removes any kept objects. When you delete an object from the keep list, all of that object's data is also deleted from the results waveform database. This also means that all occurrences of that data in any display window is removed.

To list the objects being kept, you issue the Report Keeps command, which brings up the keeps report window. It shows why each signal is being kept and whether it is kept implicitly or explicitly.

When adding information to be kept, you can specify nets, pins, and instances. If you specify one or more instances, all nets beneath the instances are kept. This can enhance debugging specific portions of your design, but can keep an excessive amount of information and result in slow performance.

You can keep a "moving window" of signal history, where only a limited amount of the most recent signal history is kept, by using the Add Keep command with the -Window switch. Windowed keep information is maintained within the kernel (not the waveform database) until the signal is displayed, thus improving performance. You use the Setup Keep button to set the window length.



- Unit-delay
  - Output and IO pins use 1 timestep delay
  - Simulator ignores all technology files
  - This is the default timing mode
- Linear Timing
  - Uses piece-wise linear approximations
- Full timing
  - Uses all technology file Rise or Fall pin delays
  - Uses BLM and VHDL delay instructions
  - No constraint checking performed
  - Complete timing accuracy with full constraint checking

## **Simulation Timing Modes**

QuickSim II has three timing modes:

- Unit-delay. Provides high runtime and invoke-time performance at the expense of accuracy and constraint checking. Use this mode when you're debugging design functionality. In this mode, all output and IO pins use a rise and fall delay of 1 simulation timestep (default is 0.1 nanosecond), and input pins use a rise and fall delay of 0. The simulator ignores all technology files, allowing for the fastest simulations. *This is the default timing mode*.
- **Full timing**. Provides complete timing accuracy without timing constraint checking. Use this timing mode to introduce timing into your simulation and for final design verification testing. This mode uses all technology file-specified delay equations, any Rise or Fall pin delays, and BLM and VHDL delay instructions. You can enable the different constraint checks any time during the simulation.
- Full timing with constraints. Provides complete timing accuracy with full constraint checking. You can use this mode to produce timing violation messages during full-circuit debugging operations. This mode uses all technology file-specified timing equations, any rise and fall pin delays, and BLM and VHDL delay instructions. This mode also checks for timing constraints and spike, contention, and hazard violations. This mode gives you the most timing and debugging capabilities (but at the greatest cost to simulator performance).

Each mode automatically sets simulation conditions that make speed and accuracy trade-offs. The higher the timing accuracy, the slower the simulation.

You can set the timing mode for the entire design when you invoke the simulator, or you can use individual commands once the simulator is invoked. Just choose the **Setup > Kernel** menu item and click on the appropriate "Timing mode" button.

## **Scaling Delays**

#### Scale factor of timing values (timing mode only) Force 1 0@t=201@t=30 0@t=50 IN 11 13 OUT3 12 0 10 OUT1 OUT 20 20 Global Scale = 0.80@t=161@t=240@t=40I2 also Scale = 1.20@t=161@t=280@t=44Choose: Setup > **Change Timing Mode** Kernel > On Selected instances Named instances Change > Timing Mode Instance name Timing Mode = unit Change... Delay Scale 1 Override OK | Reset Cancel Help

- You can apply scaling to objects on:
  - Design-wide basis -or-
  - Instance-by-instance basis (shown)
- Use delay scaling to
  - Estimate the derating effect on the simulation
  - Scale typical timing values to emulate minimum or maximum timing values

## **Scaling Delays**

When you use either of the full-timing modes (with or without constraint checking), you can specify a delay scaling factor. The simulator multiplies the delay values (pin and path delays) of the affected instances by this scaling factor.

You can scale delay values on a design-wide basis or on an instance-by-instance basis. Design-wide scaling gives you a fast and convenient estimate of how derating might affect the simulation. If you use both design-wide and instance scaling in the same simulation, the override option determines which is used.

The example show three inverters with rise and fall timing properties.

Although the default is to apply this scale factor to "typical" timing values, you can also specify that only the minimum or only the maximum timing values be scaled.



You cannot scale delays of instances that use the unit delay timing mode. Unit delay mode always uses the resolution of the simulator as the timing unit.

To scale delays for instances, click on the [Setup] Timing Mode palette button. The Change Timing Mode dialog box appears. It allows you to change the timing mode and Delay Scale for an instance or group of instances.

# **Simulation Expressions**

- Expression -- returns a single value
- Expression types
  - Arithmetic expressions use:
    - Addition "+"
    - Subtraction " "
    - Multiplication "\*"
    - Division "/"
  - Boolean expressions use:
    - ! (NOT operator)
    - o == (equality) != (not equal)
    - $\circ$  < (less than) <= (less than or equal to)
    - $\circ$  > (greater than) >= (greater or equal to)
  - Logical expressions produce:
    - True or False
    - True to False change
    - False to True change
- Expressions are combinations of:
  - Objects CLR, ENA, R\_W
  - Bus names ADDR(15:0), DATA(31:16,1,0)
  - **Constants 23.7, 3.14159, 2\*\*10,**
  - Operators (boolean and arithmetic)

## **Simulation Expressions**

A simulation expression can be as simple as a single object name, but is more often a combination of objects, waveforms, bus names, constants, and operators that returns a single value. The value of an expression is obtained by applying the operators on the operands (objects, waveforms, bus names, and constants), and the value obtained may be assigned to a variable.

Expressions can be arithmetic or logical. There are rules about the types of the operands that you can use, depending on the operators in the expression.

Arithmetic expressions can include addition, subtraction, multiplication, division and exponents. You can add or subtract operands only if they are the same type, and the result has the same type. For example, if **a** and **b** are numeric constants, then  $\mathbf{a} + \mathbf{b}$  is a valid expression that is a numeric constant.

Logical expressions deal with true or false conditions. Thus "truth tables" that determine logical operations are used to evaluate the expression. The four conditions can be used as triggers from logical expressions:

- True
- False
- True to false change
- False to true change

Logical expressions can be used as digital logic functions, that is, to produce behavior of circuitry that has not been developed. Expressions can be named and used in stimulus generation.

Expressions are useful for defining simulation conditions, such as a partial combination of signal states, for which you want to test. For example, you might use an expression as a condition for a simulation breakpoint. When the expression evaluates to TRUE, the simulation is paused. You can use expressions in many functions or commands where it is legal to use object names, because an object is just a simple, or degenerate, form of an expression.

## **Expression Examples**

Example 1: Equality, "neta" and "netb" ADD LIsts "neta = = netb"

Example 2: Bitwise AND; "busa" and 0xF3B ADD MOnitors "busa & 0xF3B"

Example 3: Boolean logic ADD BReakpoint ((d= =1r)&&(clk= =0s)) -NOC

Example 4: First & last four bits 32-bit bus "busa" CREate EXpression expr1 busa(31:28, 3:0)

To use this expression in a breakpoint: ADD BReakpoint expr1

### **Expression Examples**

• **Example 1** -- Check for equality between two nets, "neta" and "netb". To do this, activate the List window and issue the command:

#### ADD LIsts "neta == netb"

The List window displays the Boolean result of the two nets.

• **Example 2** -- Perform a bitwise AND between the value "busa" and the hexadecimal constant 0xF3B. To do this, activate the Monitor window and use an expression in the Add Monitors command.

#### ADD MOnitors "busa & 0xF3B"

• **Example 3** -- Stop the simulation only when net "d" becomes a resistive logic 1 AND net "clk" becomes a strong logic 0. To do this, issue the following command:

#### ADD BReakpoint ((d == 1r)&&(clk == 0s)) -NOChange

The -Nochange switch ensures that the simulation stops only when the expression evaluates as true. The default switch (-Onchange) would cause the simulation to stop every time the evaluation changes. The breakpoint feature is discussed in greater detail in the lesson "Using Breakpoints" on page 8-16.

• **Example 4** -- Use an expression multiple times that contains the first and last four bits in a 32-bit bus named "busa". Create an explicit expression using:

#### CREate EXpression expr1 busa(31:28, 3:0)

Now, instead of having to enter "busa(31:28, 3:0)" in subsequent commands, you can enter "expr1", such as in the Add Breakpoint command:

#### ADD BReakpoint expr1

### **More Expression Examples**

Example 5: Value (0 0 0 1 1 0 1 0) ADD TRaces databus(11, 10, 9, 8, 6, 2:0) == 0x1A Indicate bits 0 through 2 using numeric range "2:0"

**Example 6:** 32-bit bus high order bit changes ADD BReakpoint address(31)

High 4bits and the low 4bits "address" is "0xFF" ADD BReakpoint "address(31:28, 3:0) == 0xFF"

## **More Expression Examples**

• Example 5

You want a "databus" trace to resolve to a logical 1 (true) when bits 11, 10, 9, 8, 6, 2, 1, and 0 contain the value  $(0\ 0\ 0\ 1\ 1\ 0\ 1\ 0)$ , where bit 11 is the most-significant bit and bit 0 is the least-significant bit. Activate the Trace window and issue the following:

### ADD TRaces databus(11, 10, 9, 8, 6, 2:0) == 0x1A

Notice that you can indicate bits 0 through 2 using the numeric range "2:0".

• Example 6

You want the simulator to suspend execution with a breakpoint if the highorder bit of a 32-bit bus named "address" changes its logical state. To trigger the breakpoint, issue the following:

### ADD BReakpoint address(31)

Conversely, you might want to suspend execution with a breakpoint if the combination of the high four bits and the low four bits in "address" is equal to the constant "0xFF". To trigger the breakpoint under these circumstances, issue the following:

#### ADD BReakpoint address(31:28, 3:0) == 0xFF



For more information on and examples of expressions, refer to the "Simulation Expressions" section of the *SimView Common Simulation Reference Manual*. For more information on and examples of expression functions, refer to the "Expression Functions" section of the *SimView Common Simulation Reference Manual*.

## Lab Preview

- Invoke QuickSim II on a design
- Set up the environment
- Load a forcefile into "forces"
- Create, relocate, and resize windows for custom display
- Save the setup groups (including "forces")
- Exit QuickSim II
- Reinvoke QuickSim II on the same design
- Restore the setup groups (including "forces")
- Exit QuickSim II

## Lab Preview

In the lab exercise for this module, you will:

- 1. Invoke QuickSim II on the add\_det design.
- 2. Set up the SimView/UI environment using the **Setup > Environment** menu items.
- 3. Create a schematic view and Objects report windows.
- 4. Load a forcefile into "forces" using the **Setup > Force > From File** menu item.
- 5. Create Trace, List, and Monitor windows.
- 6. Relocate and resize all the displayed windows.
- 7. Save the setup groups and the "forces" waveform database.
- 8. Exit QuickSim II.
- 9. Reinvoke QuickSim II on the add\_det design.
- 10. Restore the setup groups which includes reloading the "forces" waveform database.
- 11. Exit QuickSim II.

## Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

### Procedure 1: Setting Up QuickSim II

1. If necessary, log into your workstation and set your working directory to your *qsim\_n* directory by issuing the following:



2. Invoke QuickSim II on the add\_det design by issuing the following command:



The QuickSim II session window appears.

3. Maximize the QuickSim II window.
4. Choose the following menu item to set the QuickSim II front end environment (SimView/UI), which is defined as the non-kernel configuration:

#### (Menu bar) > Setup > Environment

The Setup Environment dialog box appears (shown on page 5-10). Note the current settings and complete the following information:

- a. User Scale Type: \_\_\_\_\_ Scale: \_\_\_\_\_
- b. Write Report Page Width: \_\_\_\_\_ Length: \_\_\_\_\_
- c. Exit queries \_\_\_\_\_
- d. Default force type \_\_\_\_\_
- e. Default display radix \_\_\_\_\_
- 5. Now change the Setup Environment dialog box settings as follows:



6. **OK** the Setup Environment dialog box.

- Verify your new SimView/UI environment settings by choosing the Setup > Environment menu item again. The new Setup Environment dialog box should show the changes.
- 8. **Cancel** the dialog box.
- 9. Now create and adjust the schematic view as follows:
  - a. Click on the Open Sheet palette icon.
  - b. Maximize the schematic view window by clicking on the maximize button. You will return this window to its original size in a later step.
- 10. Report information on the D-flipflop instance by performing these steps:
  - a. Verify that all objects are unselected by using the  $|\downarrow |$  stroke.
  - b. Select the D-flipflop instance
  - c. Use the r stroke to generate a report on the selected object.

		C	)bjects	IJ	
1	<u>Sele</u> cted Instance				$\square$
/	I\$245				
	Primitive Instance	:	TRUE		
	Container Model Type	:	Schematic Sheet		
	Sheet Name	:	<pre>path/add_det/schematic/sheet1</pre>		
	Sheet Version	:	19		
	Defining Comp Name	:	<pre>path/qsim_n/lib/dff</pre>		
	Defining Comp Version	:	1		
	Number of pins	:	6		
	Properties:				

d. Do not close the Object report window, but pop it behind the schematic view window by using the stroke.

11. In this step you will load stimulus to the "forces" waveform database from a forcefile. To save you the time of retyping long lists of Force commands, a forcefile named "forcefile\_demo" has been created beneath the add\_det container.

There are two methods of loading a forcefile. The first method you used at the beginning of the Module 1 lab exercise. It involved issuing the Dofile command, followed by the path to the file. The second method uses the **Setup > Force > From File** pulldown menu item and the subsequent Navigator.

Load the "forcefile\_demo" forcefile into the "forces" waveform database by using the method of your choice.

- 12. Verify that the forces waveform database has been loaded with the forcefile\_demo waveforms as follows:
  - a. Choose: **Report > Waveforms** pulldown menu item.
  - b. In the dialog box list, select "forces (Force Target)"
  - c. **OK** the dialog box.

You should see the individual waveform names for the input forces.

In the next series of steps you will build and setup the remaining windows whose locations and sizes you want to save in the setup.

- 13. Return the schematic view window to its original size and location (the Maximize button). After the schematic view window resizes, you may need to adjust the view (the stroke).
- 14. Add all the input signals to a Trace window by performing the following steps:

a. Select all of the input nets in the schematic view window by dragging the mouse pointer as shown here:



All of the input nets are selected (change to dashed lines).

- b. While still in the schematic view window, add the selected nets to a Trace window by using the → stroke.
- 15. Add all the output signals to List and Monitor windows by performing the following:
  - a. Select all of the output nets in the schematic view window by using the same drag selection method.
  - b. List the selected nets by using the stroke.

c. Monitor the selected nets by clicking on the following palette icon.



- Display the transcript window by choosing the MGC > Transcript > Show Transcript pulldown menu item.
- 17. Resize and relocate the windows you've created so that all the signals can be viewed, but that each window uses the minimum amount of real estate. An example layout map has been provided here:

Monitor	Objects	5	Transcript		Palette
		fore	ces	List	
/:she	eet				
	Tra	ce			

18. Examine the state of the ACCESS output bus in the List window.

Why are there only 4 X's for this 16-bit signal?

- 19. Change the ACCESS bus radix to binary by doing the following steps:
  - a. Unselect all selected objects by using the stroke in the currently active window.
  - b. Select the gadget for the ACCESS bus in the List window. The gadget is the rectangular box between the signal name and its values.



c. Click on: **EDIT** > Change...

The Change List Column Attributes dialog box appears.

d. Change the Display radix to "Binary" as shown and **OK** the dialog box.



The List window now displays the ACCESS signal as a 16-bit signal.

e. Unselect the ACCESS gadget.

20. Now, because the add\_det design does not yet have the design viewpoint saved to disk, create one by entering the following command:

Click on: **DESIGN CHG** and click **YES** on the question box.

21. Next, save the setup groups to a file under the viewpoint by choosing the following pulldown menu item:

```
(Menu bar) > File > Save > Setup
```

22. When you have completed the Save Setup dialog box as shown here, click on **OK** to save the setup information.

Save Setup									
Viewpoint									
Leafname quicksim_setup									
☐ Replace									
Query when Waveform DBs have edits pending									
Save All System Setup Groups									
OK Reset Cancel Help									

The new setup information is saved, including the "forces" waveform database which is saved under the setup group.

When you restore the setup information, all the windows that were saved will have their location, size, and contents restored to their saved state. You will restore the saved setup group in the next lab procedure.

23. Exit your QuickSim II session ("Without saving" because you just did so).

#### **Procedure 2: Restoring the SimView Setup**

This procedure illustrates how the setup groups are restored and some of the information that is not restored.

1. Invoke QuickSim II on the add\_det design as you did in Procedure 1:

□ \$HOME/training/qsim_n						
shell>	cd \$HOME/training/qsim_n \$MGC_HOME/bin/quicksim add_det					

The QuickSim II session window appears in the default configuration.

2. Now, restore the QuickSim II setup groups by choosing the following pulldown menu item:

```
(Menu bar) > File > Restore Setup
```

3. When the Restore Setup dialog box appears, fill it out as follows:

Restore Setup							
Viewpoint							
Available Setups							
quicksim_setup △							
Restore setup without confirmation							
<b>Do NOT restore WDBs</b> (This supercedes manual group selection.)							
Restore All System Setup Groups							
OK Reset Cancel Help							

4. Click on the **OK** button.

A question box appears, as shown.



5. Click on the **Yes** button.

The windows and signals that you had configured at the end of the previous lab procedure are restored now. This can save considerable work in setting up similar window environments.

As the setup groups are restored, the QuickSim II session window is redrawn with the windows in their saved locations and with their saved contents. Notice however, that although the "forces' waveform database is loaded and connected as stimulus, the waveform report window is not restored; neither are the Objects or Transcript windows. Report and transcript windows are not saved or restored.

Notice that the size of the session window was not changed when you restored the setup. All window sizes within the session are scaled to the current session window size.

6. Maximize the QuickSim II session window.

#### **Procedure 3: Creating and Using Expressions**

- 1. Delete the current forces and load a new set of stimulus as follows:
  - a. Choose: (Menu bar) > File > Unload > Waveform DB
  - b. Select the "forces (Force Target)" waveform database and OK it. The "forces" waveform database is unloaded.
  - c. Issue the command:

Schematic\_view dof add\_det/lab6\_forcefile

You now have a new set of stimulus loaded into the "forces" WDB.

- 2. Create expressions for the ACCESS bus as follows:
  - a. Choose: **Add > Expression**

The CREate EXpression prompt bar appears.

b. Fill out the prompt bar as follows, and **OK** it:

CRE EX	Expression Name	EXPR1	Expression Value	access(7) == 1	ОК	Cancel	
							í.

c. Using the method in steps a and b, create the 3 additional expressions described below:

EXPR2 access(15) == 1 EXPR3 access != 0 EXPR4 (full==1s) ^^ expr2 3. Check your expression definitions as follows:

#### Choose: (Menu bar) > Report > Setup > Expressions

The following figure shows the Expression report window that is displayed.

	Expressions
Name	Expression A
expr1	access(7)==1
expr2	access(15)==1
expr3	access!=0
expr4	(full==1s)^^expr2

Notice that the expression names are in lower case, not the upper case that you entered. Expression names are not case sensitive.

What operation does the "!" symbol perform? \_\_\_\_\_\_

- 4. Save only the expressions you have created, as follows:
  - a. Execute: (Menu bar) > File > Save > Setup
  - b. In the Save Setup dialog box, enter the following and OK it.:

F	1. Click on
	Save Setup
	Viewpoint 2. Enter
	Leafname expr_setup
2 Oliok off	Query when Waveform DBs have edits pending
3. Click off	Specify the System and/or User defined setup groups to save User defined setup groups User defined setup groups Click
	OK Reset Cancel Help

5. Using the Expressions window, select and delete expr2.

What happened to expression 4?	

Why? \_\_\_\_\_

6. Add expr2 again, as you did in Step 2.

Did expression 4 return?

- 7. Restore the saved expressions, as follows:
  - a. Choose: **File > Restore > Setup**
  - b. Fill out the Restore Setup dialog box as follows, and OK it.



NOTE: The dialog box does not present you with all of the groups in a list box, because only one setup group was saved. The expression group is the only group you can restore.

8. Initialize the design to "0" by issuing the following command:



9. Run the simulator for 4500 nsec.

10. List and trace the expressions in a second Trace window as follows:

#### a. Choose: Setup > Open New Window > Trace

A new Trace window named Trace#2 is created. Because expressions do not exist on the schematic, you can't select them graphically. You can select their names in the Expressions report window.

- b. Select each of the four expression names in the Expressions report window.
- c. Activate the Trace#2 window.
- d. Click on: TRACE

The expression names appears in the Trace#2 window. Since expressions are evaluated dynamically, expression values are immediately available. A subsequent run is not required. The following shows the traces that result (use the View All stroke or key):

				Т	race					J	
	expr1 +F	+	+	+	+	+ T	+	+	+	+	
-	expr2 (F	+	+	+	+	+	+	+	T +	+	
	expr3 (F	+	+	+	+	+ T	T	TTTT	'/T +	+	
	expr4 F T	+	+	+	<b>F</b> +	+	+	+	<b>T</b> +	+	
	0.0		1000		2000 Tim	ne(ns)	3000		4000		$  \forall  $
$ \triangleleft$										$\succ$	

- 11. Examine the expression traces and answer the following questions:
  - a. Add the selected expressions to the list window using the List palette button.
  - b. Using the List window information, at what time does expr1 go true? \_\_\_\_\_\_ What time does expr2 go true? \_\_\_\_\_\_
  - c. Since access changes from all 0's to not equal to 0 only once (see List window), why are there so many transitions on the expr3 trace?
  - d. By examining the FULL signal and the expr2 trace, what boolean value must these two have to make expr4 true?
- 12. Exit QuickSim II without saving any results.



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 5 Solutions" on page A-9.

Then continue on to the next module.

### **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 5 Solutions" on page A-9.

- 1. Which of the following setups can't you report on using a Report menu item?
  - a. Actionpoints
  - b. Probes
  - c. Monitor Flags
  - d. Current Working Directory

2. Where do you set the Window Layout mode? \_\_\_\_\_

- 3. What is the default name of a saved QuickSim II setup object?
- 4. Keeping circuit activity is a function of QuickSim II operating with SimView/UI. A keep list stores signal names and their event transitions in which waveform database?\_\_\_\_\_

5. Where do you set the default force type? \_\_\_\_\_

6. How would you globally scale all timing values?

### **Module Summary**

This module shows many of the procedures required to setup a QuickSim II simulation environment.

The menu structure for QuickSim II is organized along common themes, such as reporting, editing, and setup. Menu cascading is held to a minimum. When the active window changes, or object selection changes, certain menu items are not available. You still see them, but can't select them.

The session setup is available through the **MGC** pulldown menu and allows you to specify whether some portions of the session window are hidden (to save real estate), how the windows are laid out (stacked, tiled, auto), and the type of select input device your using.

The environment (SimView/UI) setup is available through the **Setup** pulldown menu. Included in this menu are the environment, defaults, stimulus, select filter, and window attributes.

When you save a QuickSim setup, a design viewpoint is incorporated so you can save your setup object along with your user-defined waveform databases and property back annotations to this data object.

The expression syntax is powerful and AMPLE-like. Arithmetic or Logical expressions allow flexibility in how results are used. Any field that takes a value can now take an expression.

# Module 6 Design Model Initialization

Overview	6-2
Lesson	6-3
QuickSim II Logic States	6-4
Multiple Driver Resolution	6-6
QuickSim II Initialization Process	6-8
Default vs. Classic Initialization	6-10
The Initializing Process	6-12
Change Warning Start	6-14
Resetting the Simulator to Time Zero	6-16
Window Gadgets (setup)	6-18
Cross-Window Selection	6-20
Source Viewing	6-22
Using Selection Filters	6-24
Lab Preview	6-26
Lab Exercise	6-28
Procedure 1: Initializing Signal States	6-28
Test Your Knowledge	6-35
Module Summary	6-36

### **Overview**



### Lesson

Upon completion of this module, you will be able to:

- Understand how the QuickSim II simulation kernel determines resultant states on net given a set of drivers.
- Be able to initialize your design, and run a preliminary circuit initialization prior to the normal simulation run.
- Describe the difference between default and classic design initialization, and know when to use each method
- Create a source view of a selected object.
- Define a bus from a group of individual signals, and name it.



You should allow approximately 1 hour and 35 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

# **QuickSim II Logic States**

### QuickSim II is a 12-state simulator:

Drive Strength	Trace Pattern	Trace Color	Signal Level Low High		Unknown
Strong (S)		Lt Blue	0S	1S	XS
Resistive (R)		Dk Blue	0R	1R	XR
High Impedance (Z)		Green	0Z	1Z	XZ
Indeterminate (I)		Yellow	01	11	XI

- TTL designs -- require only five state values
  - 0 and 1 (for driving devices),
  - X (for either 1 or 0, but you don't know which),
  - 1R (for pullup resistors) and
  - XZ (for any high-impedance signal level).
- ECL pulldown resistors -- substitute 0R for 1R
- MOS designs -- indeterminate signal strength
- Fixed drive -- different from simple strong (S)
  - Cannot be overridden by contending signals
  - Used for voltage sources (VCC, VDD, VSS) or ground (GND)

## **QuickSim II Logic States**

The simulator uses three logic values: 0, 1, and X. The X value represents a logic value that could be *either* 0 or 1, but cannot be reliably determined. X logic values can occur at design "power-up" or as a result of signal contention, where competing logic values are driven simultaneously onto the same net.

Signal drive strengths allow the simulator to accurately resolve signal contention and to simulate subtle effects of different design technologies. The simulator uses four signal drive strengths: strong (S), resistive (R), high impedance (Z), and indeterminate (I). It combines the signal drive strengths with the three logic values to create the twelve signal states required for comprehensive and accurate simulation of the different design technologies.

TTL designs typically require only five of these logic value/signal strength combinations: 0 and 1 (for driving devices), X (for either 1 or 0, but you don't know which), 1R (for pullup resistors) and XZ (for any high-impedance signal level). By substituting a 0R in place of the 1R, you can accurately model ECL and its pulldown resistors. The need to accurately model MOS designs at the transistor level gave rise to the *indeterminate* signal strength, enlarging the state-strength table to 12 combinations. The table on the opposite page shows the resulting combination map used by QuickSim II.

QuickSim II uses an additional drive strength that cannot be overridden by contending signals, which allows you to simulate a driving positive voltage level (VCC) or ground level (GND). This overriding drive condition is a *fixed* drive, and is different from the simple strong (S) drive.

For example, when the 1S and 0S signal states are combined, the result is XS, or the unknown signal state. However, a fixed signal state of 1SF, which you would use to model a VCC connection, always overrides any other contending signal state during a simulation (stays "fixed" at 1S in this example). A fixed signal state is also useful in debugging because it cancels the effects of any driving output that is connected to the net.

	0Z	XZ	1 <b>Z</b>	0R	XR	1 <b>R</b>	<b>0I</b>	XI	1I	<b>0S</b>	XS	<b>1S</b>
0Z	0Z	XZ	XZ	0R	XR	1 <b>R</b>	01	XI	XI	0S	XS	1 <b>S</b>
XZ	XZ	XZ	XZ	0R	XR	1 <b>R</b>	XI	XI	XI	0S	XS	1 <b>S</b>
1 <b>Z</b>	XZ	XZ	1Z	0R	XR	1 <b>R</b>	XI	XI	1I	0S	XS	1 <b>S</b>
0R	0R	0R	0R	0R	XR	XR	01	XI	XI	0S	XS	1 <b>S</b>
XR	XR	XR	XR	XR	XR	XR	XI	XI	XI	0S	XS	1 <b>S</b>
1 <b>R</b>	1 <b>R</b>	1 <b>R</b>	1 <b>R</b>	XR	XR	1 <b>R</b>	XI	XI	1I	0S	XS	1 <b>S</b>
<b>0I</b>	01	XI	XI	01	XI	XI	01	XI	XI	0S	XS	XS
XI	XI	XI	XI	XI	XI	XI	XI	XI	XI	XS	XS	XS
1I	XI	XI	1I	XI	XI	1 <b>I</b>	XI	XI	1I	XS	XS	1 <b>S</b>
<b>0S</b>	0S          XS	XS	0S	XS	XS							
XS	XS	XS	XS	XS	XS	XS	XS	XS	XS	XS	XS	XS
<b>1S</b>	1 <b>S</b>	XS	XS	1 <b>S</b>	XS	XS	1 <b>S</b>					

### **Multiple Driver Resolution**

• Pessimistic Model used (X when unsure)

Example: Five net drivers: 0S, XI, XR, 1Z, and 0I

- 1. Separate categories [0S, XR, and 1Z]; [XI and 0I].
- 2. Plot XR and 0S to yield 0S
- 3. Plot 0S and 1Z to yield 0S
- 4. Plot XI and 0I to yield XI
- 5. Plot 0S and XI to yield XS

## **Multiple Driver Resolution**

Logic simulation accuracy is a direct function of the number of different signal states that the simulator can model. When more than one signal is connected to a single net, the simulator must have a means of determining an accurate result.

The table is a matrix that describes how the simulator resolves node contention between two or more output pins. To determine the state of a net connected to the outputs of two gates, locate the output state of one gate in the left column and locate the output state of the other gate in the top row; their cross-point indicates the state of the net.

When there are more than two outputs connected, separate the signal states into two categories: signals of strengths S, R, or Z, and signals of strength I. Then calculate a single state for each category as follows: plot the result of two states from the same category, then plot that result with another state in the same category. Continue combining plotted results with output states until you have a single state for each category. You calculate the actual state of the node using the final values from each category.

For example, consider five connected output pins that have the following states: 0S, XI, XR, 1Z, and 0I. Using the table, you would plot them as follows:

- 1. Separate the signal states into the two categories [0S, XR, and 1Z] and [XI and 0I].
- 2. Plot XR and 0S to yield 0S (use the result in next plot).
- 3. Plot OS and 1Z to yield OS (use the result in final plot).
- 4. Plot XI and 0I to yield XI (use the result in final plot).
- 5. Plot OS and XI to yield XS (result is the actual state of the node).

## **QuickSim II Initialization Process**

• Simulator must always know the state of each component (the design must be initialized)

### When does initialization occur?

- Invocation -- initialization occurs automatically
- Initialize command -- use during simulation
- Reset State -- similar to invoke initialization

### Two forms of initialization --

- Default initialization (V8 mode)
  - Performed when the simulator invokes
  - Compatible with System-1076 models
- Classic initialization (V7 mode)
  - Can only be performed with Initialize command
  - Compatible with pre-V8 versions

### **QuickSim II Initialization Process**

Before a simulation can begin, the simulator must know the state of each component in the design. To find this out, the simulator performs an initialization process when you invoke it. This initialization process assigns a state to each net in the design. Although this initialization occurs automatically, you can create custom initialization values.

The Initialize command allows you to initialize the circuit any time during a simulation. You can supply a unique value to the nets in your design by specifying this value with the command.

There are two forms of initialization: default initialization (or V8 type), which is always performed when the simulator invokes, and classic initialization (pre-V8 type). The default initialization scheme is compatible with System-1076 models. For compatibility with pre-v8 versions of the simulator, you can perform classic initialization. Both methods are described on the following pages.



**REFERENCE** For additional information on the initialization process, refer to the *QuickSim II User's Manual*.

## **Default vs. Classic Initialization**

- Similar to "power up" for electronic device
- Default -- must be used with System-1076 models



With Default initialization, you should run the simulator to allow init values to propagate before issuing forces

### **Default vs. Classic Initialization**

The effect of the initialization is similar to a "power up" for an electronic device. Here is the typical initialization sequence of events:

- 1. The simulator sets the initial state of all pins and nets according to the Initialize command (if specified). If the initialization is caused by the Reset State command or it is an invoke-time initialization, the simulator sets all pins and nets to the default state of Xr.
- 2. The simulator sets the state of all nets according to any associated Init properties. The net Init property values, which you specify during design creation or back annotation, override any values set in the previous step.
- 3. Depending on the type of initialization, the simulator either:
  - Default initialization -- The simulator evaluates all instances once using the states set in steps 1 and 2, and schedules output events according to delays (both pin delays and technology file propagation delays).



Unevaluated events exist at the end of a default initialization. Because these events might affect your simulation, it is recommended that you run the simulator for a short period of time before applying stimulus (for example, "Run 100").

• Classic initialization -- Using a delay of 0, the simulator propagates the initialized values through the circuit until it reaches a stable state (no events pending) or until the simulator reaches the iteration limit. The simulator does not advance simulation time during classic initialization. You can perform this type of initialization in the simulator using the Initialize command with the -Classic switch.

The difference between default and classic initialization is that, for the default, the simulator evaluates the instances only once and then uses the result for the final initialization value. The circuit doesn't have to be "stable." Another difference in the default initialization is that the simulator schedules new states according to the delay assigned to each instance, instead of using zero for the delay of every instance, as with classic initialization.

# **The Initializing Process**

### Why an Initialize Run?

- All unknown conditions are eliminated
- All quiet is achieved



## **The Initializing Process**

The first run you perform after invocation is a short run to stabilize your design. Since default initialization does not propagate any signal states through the design, this run does just that.

Another way to look at this run is as a power-on reset in your design. The important factors to consider when initializing your design are:

- All unknown conditions are eliminated in your design. In some functional models, this requires an initialize value other than Xr. Other models require you to provide "initialize" stimulus to override the unknown states.
- An "all quiet" condition is achieved. Since no stimulus (other than the initialize values) has been applied to your design, there should be no continuous activity. In other words, you should be able to issue the run command without a time limit given, and the simulator should eventually stop due to "all quiet." This means that you must eliminate all oscillating loops.

The flowchart on the previous page illustrates this process:

- 1. Invocation or a Reset state command sets default values to Xr. Any local Init property values will override the Xr state on a net or pin.
- 2. In special cases, use the Init command to set the default state to something other than Xr. For example, Init 1 sets all pins and nets to "1s".
- 3. Load the initialization stimulus. This may be needed to make sure all parts of the design are at the proper state.
- 4. Use the Change Warning Start command to disable Spike, Hazard, and constraint (Setup & Hold) warning messages.
- 5. Run for a short time or until the "all-quiet" condition is reached. If you do perform the run with a time limit, allow enough time to stabilize all circuit activity. The Run command without any parameters will allow your design to achieve an "all quiet" condition and then stop.

### **Change Warning Start**

- Suppress messages at beginning of FIRST run
- Use during initialization run.

Execute: Setup > Kernel > Change > Warning Start

	Change	e Warning Start
Enable Messages	and/or memo	ory invalidations at time
Which actions sh	ould be disa	abled until the above start time?
All	User-Speci	ified
Constraint N	lessages	Hazard Messages
Contention M	lessages	Model Messages
Spike messages		MTM invalidations
OF	Res	et Cancel Help

## **Change Warning Start**

Many times when you are initializing your design, you get a lot of spike, hazard and setup/hold violations. This is because your design has not stabilized yet. These messages are usually of no consequence to you because your design is not operating in its normal mode.

To eliminate this problem, QuickSim II provides a mechanism that allows you to suppress warnings and error messages for a period of time during a run. The Change Warning Start command allows you to suppress any or all of the normal messages that you might receive at the beginning of a simulation. As a result, initialize warnings are suppressed.

This command can be included in the dofile that you use to initialize your design, or you can interactively use this command at the beginning of the simulation run.

### **Resetting the Simulator to Time Zero**

- Simulation time set back to zero
- Design is initialized (default Xr)
- 'results' waveform database is cleared (stimulus and forces are not affected)

Reset
State
This option will reset the current simulation time
<b>back to zero and clear the 'results' Waveform DB.</b>
Save 'results' Waveform DB
☐ Viewpoint
Pathname Navigator
_ Replace

#### If you pick Viewpoint, the Pathname field and Navigator button are replaced with Leafname

## **Resetting the Simulator to Time Zero**

The reset operation allows you to return the simulation to time zero. This is useful if you want to perform multiple conditional runs on the same design without exiting QuickSim II. The reset operation does the following:

- The simulation time is set back to time zero.
- The design is initialized using the default initialization method (Xr).
- The "results" waveform database is cleared to accept data on the next run. You must save this data to disk either prior to the reset operation or by clicking on the Save option in the dialog box. The stimulus and forces that have been connected are unaffected by a reset operation.

Click on: **[palette] > Reset.** The Reset dialog box appears, which gives you several options:

- State. Selecting this option on the form will present a warning message. This tells you that your results may be destroyed if you haven't saved them. You can save the results by selecting the "Save 'results' Waveform DB" button. Results can be saved to either the viewpoint, or to a file. Placing it into the viewpoint allows you to easily find it for future simulations.
- Setup. Selecting this option resets the Setup to its initial value. Any changes you have made to user units, radix, keep lists, exit queries, and force type will revert back to the default values. Also, any changes to the timing mode, spike model, constraint modes, or simulation checking will be reset back to the default values.

# Window Gadgets (setup)

### A graphical signal object containing setup info




### **Window Gadgets**

There are times when you want to change the way information is displayed in a window. You can change the global rules that apply to all information before the window is created, or you can change the setup for individual objects within the window. The graphical device you use to modify individual window object setup is the gadget.

A window gadget that you may find useful for signal manipulation is the one associated with a signal row or column. Each signal name in a Trace, List or Monitor window has a gadget. These gadgets are the rectangular boxes that are displayed next to, above, or below the signal name. You can select the gadget in the same way you select the signal, that is, with the Select Mouse button. This selection is indicated by the appearance of a manipulation border around the area controlled by the gadget. The top figure on the previous page shows a selected gadget and the border.

To change the size of a gadget, you position the pointer on one of the gadget handles and use the Select Mouse button to drag the handle to a new location. This also works for columns of information in report windows.

You can manipulate gadgets as you do other objects, such as moving, copying and deleting. In addition, you can issue the **Edit > Change** menu item to access a change dialog box for that gadget. This is useful for changing the setup information, such as radix, after the signal has been entered in the window. The Change Trace Attributes dialog box that appears is shown on the previous page.

Changes that you make in this dialog box will only affect the field represented by the gadget you selected. All other fields (rows or columns) in the window will remain unchanged.

# **Cross-Window Selection**

- Cross highlighting
  - By name (text) or graphic selection
  - By time. View conditionals
- Source viewing. Creates window or adjusts view

Types. Schematic, Report, (VHDL with SimView/UI and simulator)



#### **Cross-Window Selection**

Representations of the same object are logically associated. For example, a net, the pin to which it is attached, its entries in the List and Trace windows, and its name in error messages are all related. Thus, if you select one representation of an object, all representations of that object are selected and highlighted. This is called *cross-highlighting*.

For example, if you select a pin name in an error message, the pin is highlighted on the design view, and the pin name is highlighted in the List and the Trace windows.

You can also use cross-highlighting when you select by time. For example, if you get a spike error message and select the time given in the message, you can issue the View Time command in the List or Trace window to view the conditions of various signals at that time. The same time label in the other windows will also be highlighted.

Use cross-highlighting when you get an error message to go directly to the Trace, List, or design representation of the trouble spot, and make changes before simulating again.



For more information on cross-highlighting, refer to "Using Cross Highlighting" in the *SimView Common Simulation User's Manual*.

# **Source Viewing**

- Access textual or graphical source of a design object
- Use source viewing to view:
  - The parent (which contains the specified object)
  - The child (contained by the specified object)

#### **Example**

If you're tracing a signal, but have no view window for the model generating the signal, you:

- 1. Select signal name in the Trace window
- 2. Choose the Open (selected) menu item
- 3. Schematic view will open and the signal's net is highlighted (since it is already selected)

# **Source Viewing**

You can access the textual or graphical source of a design object or traverse the schematic hierarchy from that object. For example, if an error message refers to an object that you're not currently viewing, source viewing allows you to view the object so you can take advantage of selection and global highlighting. You can also use source viewing to view the parent (which contains the specified object) or child (contained by the specified object) of an object.

For example, if you are tracing a signal, but don't have a view window open for the component generating the signal, you can select the signal name label in the Trace window, and choose the **Open > (selected)** menu item. Depending on the type of component model generating the signal, a Schematic or, if SimView/UI is operating with a simulator, a VHDL View will open, with the signal's net highlighted (because it is selected).



For more information on source viewing, refer to the *SimView Common Simulation User's Manual*.

# **Using Selection Filters**

#### Edit Menu (also in popup menu):



## **Using Selection Filters**

Selections filters allow you to use area selection mode to select only object of a specific type. The three key types of objects that you select are:

- Instances. This type of selection is useful for adding or changing properties associated with instances. You can also use this mode to globally update models (although the "By Property" option is more useful).
- Nets. This includes wires and buses. You can select wire nets and use the Add Bus operation to create buses in QuickSim II or SimView. You can also use area selection with this filter to select all input, output, or port nets.
- Pins. This is an easy method to select all of the pins of a particular instance. This can be used in conjunction with the Select Connected menu item to select nets connect to the pins.

You can create your own selection group that contains objects you need to select frequently during a QuickSim II session. Use the Add Group and Delete Group menu items to manipulate groups, and use the **Select** > **Group** menu item to select items in a specific named group.



When you click on items, they are added to a default (unnamed) selection group. Once you issue an operation, the selection is closed and the next selection unselects all previously selected items.

To reopen the previous selection, choose **Select > Reopen Selection**.

# Lab Preview

- Invoke on the add\_det circuit
- Initialize circuit to new value with Init command
- Perform an initialization run
- Determine if any unknowns (X) exists
- Disable warning messages.

#### Lab Preview

In the lab exercise for this module, you will:

- Initialize the design to a value other than the default initialization and examine the signal states in the Trace and List windows.
- Enable hazard checking between simulation runs, and fix hazard conditions in the circuit.
- Perform an initialization run without a stop time to determine if the simulation achieves an "all quiet" condition.
- After the initialization run, determine if any unknowns (X) exists in the design. The purpose of the initialization is to remove unknowns.
- Use the Change Warning Start command to disable simulation warning messages during the initialization period.

#### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

#### **Procedure 1: Initializing Signal States**

- 1. If necessary, log into your workstation and set your working directory to your qsim\_n directory.
- 2. Invoke QuickSim II on the design using shell invocation, as follows:



- a. Maximize the QuickSim II session.
- b. This invocation uses the default kernel setup. List the defaults.
  - 1. Timing mode: \_\_\_\_\_
  - 2. Delay scale: \_\_\_\_\_
  - 3. Simulator resolution: \_\_\_\_\_

Hint: Use the (**Menu bar**) > **Setup** > **Kernel** menu item and use the "Visible" button.

c. Cancel the Setup Analysis dialog box.

3. Examine the window signals and answer the following questions:

What is the current state of all the nets in the design?

What is the current simulation time? \_\_\_\_\_

What type of initialization was performed on invocation?

4. Run the simulation without specifying a stop time.

session run

Check the criteria for a properly initialized design:

- 1. The simulator stopped at time 102.4 due to an all quiet condition.
- 2. The List window shows that there are still X (unknown) signals remaining in the design. (The design is not properly initialized.)
- 5. Reset the simulator, as follows:
  - a. Click on the **RESET** palette button.
  - b. When the dialog box appears, click on "State" and then on "Save 'results' Waveform DB" so that it is not enabled, as shown, and **OK** it:

Reset			
State This option will reset the current simulation time back to zero and clear the 'results' Waveform DB.			
Save 'results' Waveform DB			
Setup			
OK Reset Cancel			

- 6. Initialize the design to "all ones" and run and initialization check as follows:
  - a. Issue the command:

session init 1

Examine the List window to verify that the signals shown are at the "1" state (represents 1s - strong).

- b. Run again without specifying a time.
- c. Examine the List window and Monitor window.

Does the simulation stop due to an all quiet condition?

Do any of the signals remain at X?\_\_\_\_\_ Which ones?\_\_\_\_\_

- 7. Reset the simulator, initialize the design to "all zeros" and run and initialization as follows:
  - a. Click on the **RESET** palette button and **OK** the dialog box.
  - b. Issue the command:

session init 0

Verify that the signals shown are at the "0" state (represents 0s - strong).

- c. Run again without specifying a time.
- d. Examine the List window and Monitor window.

Does the simulation stop due to an all quiet condition?

Do any of the signals remain at X?\_\_\_\_\_

- 8. Verify that no X (unknown) signals remain in your design, as follows:
  - a. Make sure that the Schematic View window is active. If not, activate it.
  - b. Select all nets in the design.



Note: This selects only nets that are shown in the current level of hierarchy. Since most of the components in this design are primitive, this is not a problem. Let's examine a non-primitive component.

c. Click on: DBG GATES



d. When the dialog box appears, select the I\$5 object and **OK** it.

The schematic view of I\$5 appears showing the internal model for a pullup component. Note that the net that is directly connected to the upper level schematic is selected, but the net between VCC and the buffer is not.



- e. Close this new schematic view window.
- f. Click on: DBG GATES

FXCEPT X

Note the Message window reports that no X signals remain.

SELECT COUNTS g. Click on:

> The messages window reports that "SELECTED: Nets = 0", among other listed selections. This also confirms that no X's remain.

- 9. Turn on timing in the simulation, as follows:
  - a. Choose: (Menu bar) > Setup > Kernel
  - b. Fill out the dialog box to resemble the following:

Setup Analysis				
Timing mode Current Unit Delay Constraint				
2. Click & edit				
Iming mode =     typ     Change     Delay Scale     1     Override				
Constraint mode Off State only Messages Override				
Spike model X immediate Suppress Override				
Hazard check  Override Spike warnings to display:				
Contention check Override Suppress 3. Click				
Model messages Override				
Toggle check      Override     Transport				
4. Click OK Reset Cancel				

QuickSim II will now run the simulation in timing mode with warnings and messages enabled.

10. Reset the state of the simulator using the palette button.

11. Load the forcefile by issue the following command:

Schematic\_view dof add\_det/forcefile\_init

- 12. Perform another initialization run, as follows:
  - a. Initialize the design to "0" using the Init command.
  - b. Run for 5000 nsec using the Run command.

NOTE: Although you could issue the run command without a time limit, and the simulation would eventually stop due to lack of stimulus, this practice is not a good idea, since infinite clock stimulus will cause the simulation to run forever.

A simulation messages window appears with several setup and hold violation messages. Notice that the last violation occurred at 18 nsec into the simulation. The violations are not important during the initialization, so in the next steps, you will suppress these warning messages until after this time.

- c. Close the Simulation Messages window.
- 13. Disable warning messages during the first 20 nsec of the run, as follows:
  - a. Choose: Setup > Kernel > Change > Warning Start

Notice that the Warning Start menu item cannot be chosen at this point in the simulation. You must first reset the simulator to time 0.

- b. Reset the simulation again using the palette button.
- c. Again choose: Setup > Kernel > Change > Warning Start

d. Fill out the dialog box as show:



This allows a warning grace period of 20 nsec, disabling only "Constraint Messages" which includes setup and hold messages.

e. Initialize to zero and run again without a stop time.

This time there are no warning messages. The design has been initialized. The next procedure uses this initialized design.

14. Exit QuickSim II without saving anything.



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 6 Solutions" on page A-11.

Then continue on to the next module.

## **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 6 Solutions" on page A-11.

- 1. Three drivers 1S, 0R, and XI are all driving the same net. What is the resultant state on the net? \_\_\_\_\_
- 2. By default, which type of initialization does QuickSim II perform upon invocation?
- 3. Why is it important to run the simulation after initialization but before issuing stimulus? \_\_\_\_\_\_
- 4. Describe how a bus appears in the Trace window for each of the following:
  - 1. Combined \_\_\_\_\_
  - 2. NOCombined \_\_\_\_\_

### **Module Summary**

Design initialization places a known state (Xr) on every signal in the design. QuickSim II initializes the design to Xr when it invokes. Two types of initialization are available. The default initialization, compatible with VHDL, only evaluates each model once but does not stabilize the circuit. Classic initialization runs the simulation until the circuit has stabilized.

The initialization process has you perform two main activities, and a third optional operation:

- Place initialization states on all nets that allows your design to come up in a known "power up" state.
- Run until there is no initialization events active in your design.
- Optionally, issue the Change Warning Start operation to remove any messages that may occur during this stabilization period.

Once the circuit has initialized, you can apply you regular stimulus, and then continue running.

Buses can be created from individual signals to help you manage the simulation information. You can create buses either combines (as a single signal) or NOcombined (individual signals remain). Even in the NOcombine mode, a bus is still operated on as a single entity.

Expressions allow you to compare and contrast simulation states and data, and raise flags when problems occur. You can use an expression with a breakpoint to stop the simulation when a specific condition is met.

You can also issue a set of actions when a breakpoint or actionpoint occurs. These actions might be useful for dumping data at a specific point in the simulation.

# Module 7 Creating and Modifying Stimulus

Overview	7-2
Lesson	7-3
Stimulus Management	7-4
Applying Stimulus as Events	7-6
The Force Command	7-8
The Stimulus Palette	7-10
Issuing Forces	7-12
Issuing Clock Forces	7-14
Stimulus Pattern Generation	7-16
Creating a Waveform Databases	7-18
Creating a Waveform	7-20
Waveform Editing	7-22
Setup Waveform Editor	7-24
Selecting Waveforms to Edit	7-26
Lab Preview	7-28
Lab Exercise	7-30
Procedure 1: Creating Waveforms	7-30
Procedure 2: Working with Waveform Databases	7-39
Test Your Knowledge	7-43
Module Summary	7-44

### Overview



#### Lesson

On completion of this module, you will be able to:

- Convert different types of stimulus to waveform database format, and convert waveform database format to forcefile and logfile format.
- Apply single or multiple force events to a waveform.
- Apply force events to create a clock.
- Edit the events in a waveform using several different methods.
- Create a waveform database and waveforms to place within it.
- Merge events from different waveform databases into stimulus for a design.
- Save the new waveforms and any waveform modifications.



You should allow approximately 1 hour and 40 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.



### **Stimulus Management**

### **Stimulus Management**

Recall from earlier modules that all forms of stimulus (forcefiles and logfiles) are converted into a waveform database as they are loaded or created. The simulator then deals directly with the waveform database, which is more efficient than interpreting Force commands. Also recall that you can make waveform databases *persistent* by saving them with the design.

If the waveform events that you require for a design simulation do not exist on disk, then you can create them from within QuickSim II (or SimView). In fact, you can create or modify both waveforms and waveform databases by using the Force command and the Waveform Editor features.

During simulation, QuickSim II maintains an accounting of all stimulus events in relation to the current simulation time: past, current, and pending. Only pending and future events can be modified; past and current events cannot. To modify past or current stimulus events, you must reset the simulation state to time zero. If you reset the simulation to time zero, the simulator maintains the existing stimulus and automatically prepares it for the next simulation run.

You may want to save the stimulus for or the results from a simulation in either forcefile or logfile format, or in the binary form. Also, you may want to use the stimulus or results with a third-party simulator that requires one of those formats for stimulus.



For more information about using or changing the form of stimulus, refer to "Manipulating Stimulus" in the *SimView Common Simulation User's Manual*.

# **Applying Stimulus as Events**

#### **Stimulus Options**

- Force commands session FORCe /PRE 1 375 Absolute
  - Create small waveforms
  - Small modifications to exiting waveforms



- Macro (AMPLE file) containing Force commands
- Create large complex waveforms

Notepad - path/training/qsim\_n/LATCH/forces\_forc (R)

 // SET USer Scale -type Time 1e-09

 // SETup FOrce -Charge

 FORCe /CLR 1 0.0 -Charge -Abs

 FORCe /PRE 1 0.0 -Charge -Abs

 FORCe /PRE 0 325.0 -Charge -Abs

 FORCe /PRE 1 375.0 -Charge -Abs



#### • V7 waveform compatibility



## **Applying Stimulus as Events**

You can *force*, or apply stimulus, to any net in the design. When you do, the simulator schedules a force event using the logic state (1, 0, X), signal strength (S, R, Z, I), and time you provide.

• Force commands (and functions). The most common form of stimulus. Force commands are automatically put into waveform database format in the Force Target waveform database. You can also issue forces by choosing menu items or palette icons. Because these forms of forces are geared towards one event on one signal at a time, Force commands are usually used to create or modify small pieces of waveform data.

Two main exceptions are the **[Stimulus palette] Add Clock** and **Pattern Generator** icons. These icons can create infinite waveforms with repeating data.

• Forcefile. An ASCII file that contains Force commands. This file can be created in a non-simulation environment, and then loaded into QuickSim II to be used within a batch-mode simulation run. Forcefiles are useful once you have determined and verified a set of Force commands. To create a forcefile, you can paste Force commands from a transcript, enter them manually, or use AMPLE looping constructs (identical to C).

AMPLE is the most common method for developing large amounts of simulation stimulus. You can write high level functions or dofiles to be executed in succession to drive the simulation.

Forcefiles can be applied directly to simulations. To do so, either issue a Dofile command specifying the forcefile or load the forcefile using the **Setup > Force > From File** pulldown menu item.

• **Logfile**. An ASCII file that the simulator produces or that a Pre-V8 Mentor Graphics simulator produced. Logfiles can be used by third party tool users as a method of exchanging simulation data.



For more information about the logfile format, refer to "Simulation Logfiles" in the *SimView Common Simulation User's Manual*.

# The Force Command

- Places Force Events in a waveform
- Issuing Force commands

Force a Pin, Net, or Waveform to a State at a Time

Examples: Force PRE 0 (at the current time) Force D 1 100

- Force types
  - -Charge force is active until *any* force overrides
  - -Wired force contends with other outputs
  - -Fixed force cannot be overridden
  - -Old like -charge but requires stronger force to override
- Issuing clock forces (-Repeat switch)
  - Set Clock Period command determines repetition frequency
  - Force signal state time -repeat
  - Example: Set Clock Period 100 Force CLK 0 0 -repeat Force CLK 1 50 -repeat

### **The Force Command**

Force commands (and functions) are the most common form of stimulus. When you issue a Force command you are forcing a signal to a specific state at a specific time. These are commonly referenced as forced events. When using QuickSim II, Force commands are interactive. That is, as you run the simulator you can force events relative to the current simulation time.

To force a signal named D to a state of 1 (strong) at 100 time units from the current simulation time (relative), issue the command:

Force D 1 100 (relative is the default time)

To force the same signal D to a state of 0 (strong) at absolute time 100, issue the command:

#### Force D 0 100 - Absolute

The Force command has a -type switch that specifies how the force behaves. There are four types of behavior:

- -Charge (default). The force is issued to the signal but is removed when any other force or driver is issued to the signal.
- -Wired. The force contends with all other forces and drivers of the signal.
- -Fixed. The force "fixes" the signal at that state. No other signal can change the state value of that signal.
- -Old. This force option is similar to the -Charge option except that it is only removed by a stronger force or driver to the signal. Once it is overdriven, it does not return (as does the -Wired force) when the stronger force is removed.

The Force command also has a -repeat option that allows you to issue cyclical forces (clocks). This option is used with the Set Clock Period command. To force a signal named CLK to repeat a 50% duty cycle clock at a 100 nsec period, issue the commands:

Set Clock Period 100 Force CLK 0 0 -repeat Force CLK 1 50 -repeat

# **The Stimulus Palette**

- Add, Modify, Delete Stimulus
- Load, Save, Connect Stimulus



## **The Stimulus Palette**

The Stimulus palette is displayed when you click on the **stimulus** button in the palette selection area. The icons in this palette perform several of the general stimulus management functions as well as several Force command functions. Many of the menu items contained in the (**Menu bar**) > **Setup** > **Stimulus** area are duplicated for your convenience in the Stimulus palette.

Most of the icons are optimized for use on a selected waveform, pin, net, or bus object. If either too many objects or the incorrect type of object is selected, then the icon displays the appropriate dialog box for your edification.

The icons displayed at the top of the icon area represent logical groups of Force commands that facilitate adding events or groups of events to a waveform, pin, net, or bus.

The icons at the bottom of the icon area provide the means to manage stimulus information (load, save, connect, create waveforms and waveform databases).

The two icon in the center of the palette, Nearest and Deltas, provide a way for you to easily determine exact values of events displayed in a Trace window. These two features are discussed later in Module 9, "Examining Results and Waveforms".

The palette menu helps you manipulate (show, hide, resize, and traverse) palettes.



For more information about the function of each icon button in this palette, choose the (**Menu bar**) > **Help** > **On Palettes** > **Stimulus** menu item.

### **Issuing Forces**

- To add Force at Current Simulation Time
  - a. Select waveform, pin, or net
  - b. Click on: STIMULUS

State Value	
<u>1</u> (True)	$\triangleright$
<u>0</u> (False)	$\triangleright$
X	$\triangleright$
Z	
Other States	$\triangleright$
User Specified	

- OR -
- To add One or More Forces at Any Time
  - a. Select waveform, pin, or net
  - b. Click on: **STIMULUS**



FORCE TO

c. Complete the Dialog Box

Force Multiple Values		
Signal Name   Value   Time   Value	Force type Charge Charge Absolute	
OK Reset Cancel	Help	

# **Issuing Forces**

Besides using the Force command described earlier, you can use icons in the Stimulus palette to forces state values to a waveform, pin, or net. The icon you use depends on how many events you want to force and at what time you want the events applied.

To issue single, interactive, forces at the current simulation time select the waveform, net, or pin and click on the **[Stimulus palette] Force To State** icon and choose the state and strength from the icon popup menu that appears. The state value you specify is applied directly to the selected waveform, pin, or net at the current simulation time. There is no offset, start time, or stop time options available. The force type is taken from the environment setup discussed in Module 5, "Setup and Expressions"

To issue single or multiple forces at any time beyond the current simulation time select the waveform, net, or pin and click on the **[Stimulus palette] Add Force** icon. Then enter in the dialog box the state value and time pair for each event. The state values are applied to the deleted waveform, pin, or net at the times that you specified.

Other options you can set in the dialog box are the force type and whether the times are absolute or relative. The "Force type" entry allows you to specify the way the force behaves in the circuit. The "Default" force type is "Charge". Force types were discussed in the lesson "The Force Command" on page 7-9. You can also specify whether the times you enter are absolute or relative to the current simulation time. If you click on the "Absolute" button, the time values are evaluated as relative to absolute time zero.

If you specify a time that is prior to the current simulation time, the force is not allowed; you must reset the simulation state and reissue the force. Information on resetting the simulation state will be discussed in Module 6, "Design Model Initialization".



#### **Issuing Clock Forces**

Most complex designs use one or more clocks to provide synchronous timing to the logic. A clock force is a repeating state change pattern on a signal. Most microprocessors require a clocking scheme that synchronizes the internal microprocessor timing to external devices.

The frequency of the repeating pattern is called the *period*. Each time you issue a new clock using this operation, you can enter a new period, and thus change the current period. That period remains active and is used by any subsequent command that uses the "Repeat" option.

The *duty cycle* is a measure of the time that the clock is in the active state. It is usually indicated as a percentage. By specifying the active "Time" (and value) and the inactive "Time", you are setting the duty cycle.

For example, the figure shows a clocking scheme required by a microprocessor. This clock is a two-phase clock (two signals, one period). The period is 100 for both clock phases. You must issue this operation twice, once for each signal, to define this clock. To issue a clock type force:

- 1. Select the MPCLK1 net.
- 2. Click on the [Stimulus palette] Add Clock icon.
- 3. Fill in the Force Clock dialog box as it appears in the previous page.
- 4. **OK** the dialog box.

You can alternately use the **Set Clock Period** command and **Force -Repeat** commands to generate clock type forces. Here is an example.

SET CLock Period 100 FORCe MPCLK2 1 50 -Repeat FORCe MPCLK2 0 75 -Repeat



For more information on issuing force-type stimulus, refer to "Manipulating Stimulus" in the *SimView Common Simulation User's Manual*.

### **Stimulus Pattern Generation**

001 110 010 101 100 011

PATTERN

#### **Create special stimulus patterns**

Click on:



GENERATR				
Pattern Generator				
Specify the type of pattern you want generated:				
0000         0100         0001         1110         101           0001         0011         0010         1101         010	0			
0 0 1 0         0 0 1 0         0 1 0 0         1 0 1 1         1 0 1           0 0 1 1         0 0 0 1         1 0 0 0         0 1 1 1         0 1 0	0 1			
	0			
Incr/Decr Value Walking 1 Walking 0 Alterna	ting 1-0			
Signal to generate pattern for	_			
Initial value 0 Incr Decr eac	h pattern by 1			
Radix of input values: 🐟 🔿 🕎				
Hex Octal Binary Decimal Float Signed				
Start at time 0 Duration of each pattern A Duration of each pattern				
	Default     Eived			
Total number of patterns				
	Wired			
Duration of high-impedance regions	Charge			
before and after each pattern:	✓ Old			

After

OK

0

Reset

Cancel

Clear old forces

Before 0

### **Stimulus Pattern Generation**

Sometimes your design does not require unique stimulus patterns, but only repetitive patterns, such as in automated test. The SimView/UI in QuickSim II provides a stimulus pattern generator that can help you create stimulus when repetitive patterns are needed.

You access the pattern generator from the Stimulus palette. Click on the **[Stimulus palette] > Pattern Generator** icon. The Pattern Generator dialog box appears, which allows you to specify the type and timing values for the pattern. The dialog box is shown on the previous page.

You can choose between five basic patterns:

- **Incremented value.** With this pattern you specify the "Initial Value" and the "Incr each pattern by" value. SimView/UI then continues to increment successive values.
- **Decremented value.** With this pattern you specify the "Initial Value" and the "Decr each pattern by" value. SimView/UI continues to decrement successive values.
- Walking 1. SimView/UI shifts a 1 bit-wise through a field of zeros for each new pattern.
- Walking 0. The same as for a walking 1, except 0 is shifted in a field of ones.
- Alternating 1-0. This pattern, sometimes referred to as the 5/2 pattern, forces every other line to the opposite state at each event.

The "Initial Value" and "Incr/Decr each pattern by" entry fields only appear when the Incr/Decr Value button is chosen. You must supply a value for all entry fields shown, or else you will not be allowed to **OK** the dialog box.

When you **OK** the dialog box, SimView/UI builds the stimulus in the Force Target waveform database. You can examine or edit these waveforms using techniques you learn in this module.

# **Creating a Waveform Databases**

Create user defined waveform databases



- Optionally, ease waveform editing by defining as Force Target with:
  - Click on: STIMULUS
     Set WDB Defaults
     Waveform names are referenced from: forces (Force Target) forces2
     results (Default)

 results (Default)

 stimulus
 7

 Forces are sent to and deleted from:

 forces (Force Target)

 Force Target
 1

 forces2
 1
### **Creating a Waveform Database**

In addition to the predefined waveform databases created by QuickSim II, you can create your own waveform databases.

Reasons for creating a waveform database:

- Keep waveforms separate from those in the forces waveform database.
- Keep tested waveforms separate from those still in development stage.
- Keep code fragment stimulus grouped.

If a waveform database is created to contain stimulus waveforms, you must connect it to the design manually. Unlike the forces waveform database, userdefined waveform database do not automatically connect to the design. Once connected, however, the waveforms in the user-defined waveform database are merged with any other stimulus connected to the same design object.

If you plan to edit the waveforms in a user-defined waveform database, you can optionally define it as the Force Target. This will redirect all future Force and waveform edit commands to this database, including the Create 12state Waveform command.

## **Creating a Waveform**

Create an empty 12-state stimulus waveform with:



Once created:

- Add Events using any Stimulus Palette icons
- Modify Events using any Waveform Editor icons

### **Creating a Waveform**

From within QuickSim II you can create empty waveforms and provide them names either based on selected nets, or by manually providing unique names for each waveform. To create a waveform click on the **[Stimulus] Create A Waveform** palette icon.

Creating your own waveforms in this manner can be useful during both the stimulus creation and debug analysis phase of simulation. You can use the waveforms you create as stimulus by *connecting* to a net in the design, as expected results for comparison against actual values, or as a scratch pad to specify events or times of particular interest.

You can create new waveforms in any waveform database you specify except stimulus or results.

You can only create single-bit waveforms; no buses.

## **Waveform Editing**

Wavefor	m Editor
SETUP	STIMULUS
WF EDITOR	DESIGN CHG
DBG GATES	DBG VHDL
ANALYZE	DBG HIER
TRACE	LIST
TRACE	LIST
DELETE	EDIT 🕨
UNSELECT	SELECT COUNTS
PL-	
	SETUP
WAVEFORM	EDITOR
TOGGLE	PULSE
<u> </u>	
NEAREST	DELTAS
	$\overline{X \mathbf{F} \rightarrow \mathbf{E}} $
ADD	CHANGE
MOVE	SHIFT
5 Je	
CUT	COPY
PASTE	DELETE
CLEANUP	

### **Waveform Editing**

The Waveform Editor palette is displayed when you select the WF EDITOR button. To edit a waveform using the graphical waveform editor, you must first include it in the Trace window. This can be done using several methods. One of the more commonly used methods when getting ready to edit is the Edit Waveform icon which is available in both the Waveform Editor and Stimulus palettes.

Most of the icons contained in the Waveform Editor palette operate using the following method:

- 1. Click on the edit icon of your choice.
- 2. Point and click or click and drag the mouse pointer or cross-hairs within the confines of the waveform to be edited, whichever is appropriate for the icon.

In this way you can add new events, toggle events, move events, copy or cut events from one waveform and paste them to another, and much more.



For more information about the function of each icon button in this palette, choose the (Menu bar) > Help > On Palettes > Waveform Editor menu item.

## **Setup Waveform Editor**



Setup Waveform Editor							
Snap edges to time grid							
Time grid spacing (>=1) 10 Time grid offset 0							
The Edit Waveform option requires selected signals or waveforms. If signals are selected their waveforms will be retrieved from the highlighted WDB below. If no waveform exists, a new one will be created.							
WDB to find/create waveforms for editing:							
forces (Force Target)							
Selecting WDB other than the current force target will cause the session-wide force target to be changed.							
Initial values for new waveforms							
12 State: Xr Change Bit: 0 Toggle							
9 State: U Change Boolean: False Toggle							
4 State: X Change							
OK Reset Cancel							

### **Setup Waveform Editor**

The waveform editor has characteristics that have been preset when you enter editing mode. For example, when you add or move a signal transition, the edge aligns with the nearest snap point. As the dialog box on the previous page shows, the default for this characteristic is 10 nanoseconds. Here is a list of characteristics:

- Snap edges to time grid. Enables or disables the placing of edits on the nearest snap point.
- **Time grid offset.** Allows you to offset the snap point increment from zero. For example, an offset value of 5 paired with the default grid spacing of 10 would place snap points at 5, 15, 25, and so on.
- WDB to find/create... This determines in which waveform database the waveform editor is going to look for the waveform to be edited. If the waveform in the Trace window is specified using a complete waveform name (including the waveform database prefix), then this setting is not used. However, if the waveform in the Trace window is specified only by a signal name, then you must be sure the proper waveform database is being accessed. You can edit any waveform, except those that reside in the "stimulus" or "results" waveform databases.
- **Initial values for new waveforms.** You can set the state that is used for the signal when it is first created (initialized). If you click on the 12-state button, you are presented with a selection box containing the 12 valid states. Click on the state that you want to use to initialize new waveforms.

You can change the setup at any time during a waveform editing session. This allows you flexibility in creating or editing signals that need to be synchronized to different clocks. Changing the setup does not affect previously created signals or transitions.



For more information about the setup of the waveform editor, refer to the *SimView Common Simulation User's Manual*.

### Selecting Waveforms to Edit

- Edit any waveform displayed in the active Trace window. No further action required.
- If the waveform doesn't exist or has not yet been traced:
  - a. Select the pin, net, or waveform to be edited
  - b. Click on: WF EDITOR



The waveform is created in the specified Waveform Database and entered in a Trace window

Displayed waveforms in Trace window appear as:

waveform\_database\_name@@signal\_name

**Examples**: forces@@clk 1 my\_wdb@@sig\_p

## **Selecting Waveforms to Edit**

The Edit Waveform icon is located in both the Waveform editor and Stimulus palettes. This icon enables you to quickly create and display new waveforms, or to display existing waveforms. If the waveform to be edited is already displayed in a Trace window, the icon is not necessary.

However, if the waveform target is not traced, you must select the pin, net, or waveform that you want to display or edit before clicking on the **Edit Waveform** icon. If you don't, you will get a message:

### "Selected signals or waveforms are required for the 'Edit Waveform' option."

When a pin or net is selected and an associated waveform is created and displayed, it appears in the following form:

Waveform\_database\_name@@signal\_name

This allows you to easily differentiate these objects from the waveforms that are generated by the simulation run ("results"). You can manipulate these signals just as you would a "results" waveform.



For more information about accessing or creating waveforms to be edited, refer to the *SimView Common Simulation User's Manual*.

## Lab Preview

Within QuickSim II you will:

- Open a schematic view window
- Create waveforms for the design signals and apply force events
- View the waveform events in the Trace and List windows
- Generate reports on waveform activity
- Create a waveform database and waveform
- Merge waveforms using connect
- Create a second Trace window for stimulus
- Save a waveform database to a file
- Save the "stimulus" waveform database to a forcefile
- Exit QuickSim II

### Lab Preview

In the lab exercise for this module, you will:

- 1. Invoke QuickSim II on the add\_det design.
- 2. Open a schematic view window.
- 3. Create waveforms for the design input signals, including a clock.
- 4. View the waveform events by using the Trace window.
- 5. Generate waveform and waveform database reports on waveform activity.
- 6. Create a waveform database containing a waveform.
- 7. Merge two waveforms from different waveform databases into one stimulus waveform by using the connect feature.
- 8. Create another Trace window to view the merged waveform events (stimulus).
- 9. Save a waveform database to disk, under the design viewpoint.
- 10. Save the forcefile format of the "stimulus" waveform database to disk.
- 11. Exit the QuickSim II session.

### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

### **Procedure 1: Creating Waveforms**

In this procedure you will create and manipulate waveforms using QuickSim II. You will use the graphical waveform editor to add and change waveform transitions, and you will connect user waveforms.

- 1. If necessary, log into your workstation and set your current directory to your *qsim\_n* directory.
- 2. Invoke QuickSim II on the add\_det design by issuing the following command:



The QuickSim II session window appears.

3. Create the schematic view window using any method.

- 4. Create editable waveforms for the input nets TEST, PULSE, ANALOG\_OUT, and LATCH by performing the following:
  - a. Select the following nets in the schematic view window:

TEST ANALOG\_OUT PULSE LATCH

b. Create the waveforms and display them in a Trace window by clicking on:



A waveform for each of the nets is created in the "forces" waveform database (Force Target default) and is displayed in a new Trace window. Notice that the waveform names all begin with "forces@@/" and that their values are Xr (unknown-resistive). This is the waveform editor's default initialization value.

- 5. Create editable waveforms with initial values of "1s" for the input nets START and \_CLR by performing the following:
  - a. Click on the following palette icon to change the waveform editor's default initialization value from Xr to 1s:



The Setup Waveform Editor dialog box appears as shown in the lesson "Setup Waveform Editor" on page 7-24.

b. Change the initial value for any new waveforms by completing the Setup Waveform Editor dialog box as shown here:



- c. Now select the START and \_CLR nets in the schematic view window.
- d. Create the editable waveforms and display them in the Trace window by clicking on:



What are the names and values of the six waveforms shown in the Trace window?

	Waveform Name	Initial Value
1		
2		
3		
4		
5		
6.		

6. Add isolated events to TEST, ANALOG\_OUT, and LATCH by issuing the following Force commands, recalling that to obtain the command line, just begin typing the command:

#### FORCe TEST 0 10 FORCe ANALOG\_OUT 0 0 FORCe LATCH 0 25

The Trace window is updated to show the new events in the waveforms. Notice that the forces@@/ prefix was not needed in the previous Force commands because the Force Target is already set to the forces waveform database.

You may also notice that the initial value for ANALOG\_OUT was just changed. A waveform can be initialized by the Edit Waveform palette icon using the waveform editor setup only once (at creation). As was done here, you can reinitialize a waveform at time zero using the Force command.

- 7. Add 50 nsec pulses to the START and \_CLR waveforms by performing the following:
  - a. First set the Waveform Editor's snap resolution to 5 nsec by performing these steps:

EDITOR

i. Click on the WF EDITOR

SETUP palette icon.

ii. Set the Time grid spacing (snap resolution) to 5 nsec:

Setup Wavefo	rm Editor
Snap edges to time grid	
Time grid spacing (>=1) 5	Time grid offset 0

- iii. **OK** the dialog box.
- b. Click on the WF EDITOR palette icon (Trace window active)

The Pulse prompt bar and a set of Trace window cross-hairs appear.

c. Move the cross-hairs within the START waveform at approximately the 200 ns time.

s@@/ANALOG_OUT	- +	+	+	+	+	+	+	+
200ns forces@@/START	- +	+	+	+	÷	+	+	+
forces@@/_CLR	- +	+	+	+	+	+	+	+
0.0	300.0		600.0 T	ime(	900. ns)	0	1200	.0
Pulse Locations	OK Cano	el						

d. Click and drag the Select mouse button to approximately time 250 ns while remaining within the START waveform.

s@@/ANALOG_OUT	+	+	+	+	+	+	+	+	+
2 forces@@/START	:00ns +	249ns	+	+	+	+	+	+	+
forces@@/_CLR	+	+	+	+	+	+	+	+	+
0.	0	300.0		600.0 T	) Time(	<i>900.</i> ns)	0	1200	.0
Pulse Locations		K Cano	el:						

- e. Release the Select mouse button to complete the START pulse. Notice that the pulse "snaps" to the nearest 5 ns value and that the pulse cross-hairs are ready for the next pulse.
- f. Now add a 50 ns pulse to the \_CLR waveform from 100 ns to 150 ns.
- g. **Cancel** the Pulse prompt bar.

- 8. Add events to the ANALOG\_OUT and LATCH waveforms by performing the following steps:
  - a. Using the scroll bar at the bottom of the Trace window scroll to time 2600.
  - b. Click on the WFEDITOR palette icon.

The Toggle prompt bar and a shadow cursor appear.

OK Cance

c. Move the shadow cursor within the ANALOG\_OUT waveform to approximately the 2600 ns time.

forces@@/TEST	+	+	+	2600	+ ).0ns	+	+	+	+	+	
s@@/ANALOG_OUT	+	+	+	0	Т	+	+	+	+	+	
forces@@/LATCH	+	+	+	-	+	+	+	+	+	+	
forces@@/START	+	+	+		+	+	+	+	+	+	
22	00.0	2400	.0	2600.0 Tir	<i>2</i> ne(n	800 .s)	.0	3000	.0	3200	$\nabla$
										$\triangleright$	
				-							

d. Click the Select mouse button.

Toggle Locations +

The ANALOG\_OUT waveform now transitions from 0s to 1s at time 2600 ns. Also, the Toggle prompt bar remains and is ready to insert another event.

- e. Toggle the LATCH waveform at time 2500 ns. (If the time is not visible in the Trace window, you can scroll the Trace window by using the shadow cursor to bump the edge of the Trace window.)
- f. **Cancel** the Toggle prompt bar.

- 9. Add a clock to the PULSE waveform by performing the following:
  - a. Use the  $|| \downarrow |$  stroke in the Trace window to unselect all the objects.
  - b. Select PULSE in the schematic view window.
  - c. Click on the **STIMULUS** palette icon. The Force a Clock Signal dialog box appears.
  - d. Enter the following information:

Force a Clock Signal					
Signal name /PULSE	Period 100 Stop ti	me			
Clock should have Single Tra With a Duty Cycle of 25% Clock is active High Lov	Multiple Transitions 50% 25% Other w PERIOD Clear old forces	Force type Cellon Default Fixed Wired Charge Old			
OK Reset Cancel					

e. **OK** the dialog box.

Notice the repeating clock pattern for the PULSE waveform shown in the Trace window.

- 10. Create another waveform database by performing the following:
  - a. Report the current waveform databases by choosing the following menu item:

#### (Menu bar) > Report > Waveform DBs

The Waveform DBs report window is displayed. Note which waveform databases are currently listed in the Waveform DBs report window. (This step is only performed to help illustrate the creation of a new waveform database.)

b. Create a new, unique, waveform database by clicking on the following palette icon:



The Create Waveform Database prompt bar is displayed.

c. Enter the following in the WDB Name entry area:



The Waveform DBs report window is updated to show the new waveform database and the following message is displayed in the message area.

### A new Waveform Database named 'forces2' has been created.

- 11. Create a new waveform in the forces2 waveform database for the ANALOG\_OUT signal by performing the following:
  - a. Set the Force Target to forces2 by clicking on the following palette icon:

STIMULUS	
----------	--

The Set WDB Defaults dialog box appears.

b. Select "forces2" and **OK** the dialog box:

	Forces are sent to and deleted from:
	forces (Force Target)
Force Target	forces2
	OK Reset Cancel

Note that the Waveform DBs report window is updated to show the new force target.

- c. Select the ANALOG\_OUT net in the schematic view window.
- d. Click on the WF EDITOR palette icon:

The new waveform with the prefix "forces2@@" is added to the Trace window.

e. Edit the new waveform using the **WF EDITOR** palette icons to add (use the ADD palette button) a 0s event at 4100. Be sure to cancel any prompt bars when your finished.

### **Procedure 2: Working with Waveform Databases**

Now that you have a group of waveform events for the design, you will examine how these events will be presented to the design for simulation and then save the waveform events for use by a simulator.

- 1. Examine the overall stimulus before saving for a simulation session by performing the following steps:
  - a. Create a second Trace window by choosing the following menu item:

### (Menu bar) > Setup > Open New Window > Trace

A new Trace window appears. What is this window named? \_\_\_\_\_

- b. Move the new Trace window vertically in the session window so you can view both Trace windows.
- c. Report the waveforms connected to the design as stimulus by performing the following steps:
  - i. Choosing the following menu item:

### (Menu bar) > Report > Waveforms

The Report Waveforms dialog box is displayed.

ii. Select the "stimulus" waveform database and **OK** the dialog box.

The stimulus report window is displayed.

- d. Trace the contents of all waveforms connected to the design as stimulus by performing the following steps:
  - i. Select each of the waveforms in the stimulus report window. (Hint: select the waveforms in the same order as they appear in the original Trace window. This will make the comparisons easier.)
  - ii. Use the  $\neg$  stroke to add the selected waveforms to the Trace#2 window.

The Trace#2 window is updated with the stimulus waveforms.

e. Examine the contents of the two Trace windows to ensure that all the stimulus you created is in the stimulus waveform database.

Notice that the events in forces2 for signal ANALOG\_OUT at time 4100 is not present in the stimulus waveform database. This is because only the waveforms in the "forces" waveform database are connected by default. You must manually connect any others.

- 2. Connect the waveform events in the forces2 waveform database as stimulus by performing the following:
  - a. Click on the following palette icon menu item to pop the Waveform DBs report window to the foreground:



- b. Select the "Waveform DBs" text and **OK** the dialog box.
- c. In the Waveform DBs report window, select the forces2 waveform database.
- d. Choose the (Waveform DBs) > Connect popup menu item.
- 3. Examine the contents of the two Trace windows again. The waveform events from forces@@/ANALOG\_OUT and forces2@@/ANALOG\_OUT have been merged into one waveform: stimulus@@/ANALOG\_OUT.

- 4. Save the "forces" and the "forces2" waveform databases to the design viewpoint by performing the following steps:
  - a. Click on: **STIMULUS**



b. Select the "forces" waveform database and fill out the remainder of the dialog box as follows:

Save Waveform DB
Waveform DB
forces
forces2 (Force Target) results (Default)
2. Click
Viewpoint 3. Fill in
Leafname forces_lab7
File type WDB Logfile Forcefile
Replace 4. Click on
OK Reset Cancel Help
5. Click OK

- c. Save "forces2" by using the same method but, provide this different pathname: **forces2\_lab7**.
- 5. Save the "stimulus" waveform database information, thereby saving all the merged waveform information by issuing the following command line:

```
    Waveform_DBs
    SAVe FOrcefile stimulus_lab7 stimulus -STOp 5500 -Viewpoint
```

The "stimulus" waveform database is saved to the design viewpoint in forcefile format. The "stimulus" waveform database is unique; you can't save it using Save Wdb. This is, in part, because the stimulus waveform database exists only within the confines of QuickSim II and is really just a collection of all the waveform databases *connected* to the design as stimulus.

6. Examine the *stimulus\_lab7* file (beneath the "default" viewpoint) using the Notepad editor in read-only mode.

Notice the setups that are created at the beginning of the file. The Absolute switch has been added to all Force commands. Also, the PULSE clock signal is now a finite series of individual forces.

Close the Notepad editor when you are finished.

7. Exit QuickSim II by issuing the following in the popup command line:



You are given one last chance to save information.

8. Verify "Without saving" is selected, then **OK** the dialog box.

No further objects are saved and QuickSim II exits.



### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 7 Solutions" on page A-13.

Then continue on to the next module.

### **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 7 Solutions" on page A-13.

- 1. When you issue the Force command, what waveform database is the force entered into, by default?
- 2. When you use the Waveform Editor palette, what waveform database default mechanism determines the waveform database into which the edits are entered?
- 3. What palette icon would you use to change the initial event value for a new waveform about to be created?
- 4. In terms of a QuickSim II menu item, palette icon or command, what does "make a waveform database persistent" mean?
- 5. What is the menu path for online help information about the individual palette icons?

## **Module Summary**

In Module 7 you learned how to use the palettes to create and modify waveforms. More information about each of the palette icons is provided by choosing the **Help > On Palettes > Palette Descriptions** menu items.

The Stimulus palette provides icons that allow you to add and delete individual forces as well as clocks. This palette also provides icons to create, load, and save waveform databases.

The Waveform Editor palette allows you to graphically manipulate waveform events. You can add waveforms from the "force target" waveform database to a Trace window to be edited. Waveform transitions can be inserted, copied, inverted, or deleted.

# Module 8 Running a Simulation

Overview	8-2
Lesson	8-3
Performing a QuickSim II Run	8-4
QuickSim II Analysis Evaluations	8-6
Timing Wheel Concept	8-8
Iterations	8-10
Timing Accuracy	8-12
Setting Run Parameters	8-14
Using Breakpoints	8-16
Using Actionpoints	8-18
Saving and Restoring Simulation States	8-20
Batch Simulations	8-22
Setting Up Batch Simulations	8-24
Running a Batch Simulation	8-26
Lab Preview	8-28
Lab Exercise	8-30
Procedure 1: Evaluations and Oscillations	8-30
Procedure 2: Resolution and timing	8-38
Test Your Knowledge	8-42
Module Summary	8-43

### **Overview**



### Lesson

Upon completion of this module, you will be able to:

- Describe how QuickSim II performs a simulation run, using the terms iteration, scheduled event, and mature event, and timing wheel slot.
- Describe how a hazard warning can occur, and what can be done to prevent a hazard.
- Describe how an oscillation can occur in a digital design, how to prevent oscillations from occurring, and use the command to change the oscillation limit.
- Describe the difference between a breakpoint and an actionpoint, and be able to create a breakpoint.
- Save and restore simulation states.
- Reset the simulator back to time zero without saving the current results waveform information.
- Setup and run a simple batch simulation, using a previously created stimulus waveform database, and a batchfile redirected to control the simulation.



You should allow approximately 1 hour and 20 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

# Performing a QuickSim II Run

Run pulldown menu:



Run Menu	Description
Step	For VHDL debug. Steps over statements, into statements, into iterations, until an event occurs, or until the end, or resumes a paused simulation.
Activate	VHDL. Activates the named or selected statement.
Simulation	Runs simulation for a duration of time, until a specific time, or until it stops, or resumes a paused simulation.
Initialize	Uses the default method to initialize the design.
Reset	Resets the state of the simulation, with or without saving the results WDB.

Better yet, use the run command:

- Run for time: run 1000
- Run until time: run 1000 -a
- Run until stop: run

## Performing a QuickSim II Run

The run command starts the simulation kernel stepping through an analysis. The kernel determines if there are any task to do, and then performs them in a logical progression.

Although you can use the palette button, menu item, or several other methods to start a simulation run, the run command is probably the quickest to issue. The options you can use with the run command are:

- Run for time: Run 1000
- Run until time: Run 1000 Absolute. With the Absolute command switch, the time value is interpreted as an absolute value. This simulation will stop at that time. If the time is a past value, QuickSim II issues a warning message and the run is not performed
- Run until stop: Run. The simulation runs until:
  - A breakpoint condition is met. This is considered a pause which allows the simulation to be restarted by the "Resume" command.
  - You press the stop key. On the HP-Apollo workstation, pressing the Ctrl-S key will stop the simulation. On the other UNIX workstation types, pressing the Ctrl-C key stops the simulation.
  - An "all quiet" condition occurs. All quiet means that there are no current events to be processed, and there are no events pending (no stimulus).

# **QuickSim II Analysis Evaluations**



- Event -- Change of state at a given time
  - Evaluating event (mature)
  - Scheduled event (delays)
- Scheduling method -- "Timing wheel" algorithm

Note: Devices such as the Pullup resistor do not schedule events. Therefore, you must "kickstart" circuits using these devices to change states.

### **QuickSim II Analysis Evaluations**

Understanding how the simulator processes a simulation can help you use it more efficiently. The following text describes the type of circuit activity the simulator processes, and how it schedules the activity for processing. The discussion begins with some basic concepts and then covers the detailed QuickSim II kernel scheduling algorithm.

The basic circuit activity that the simulator processes is called an *event*. An event occurs when a signal state changes (either the logic value or the drive strength). These state changes can be caused by a component's output or by input stimulus you provide to the design. The figure illustrates a simple example of two events.

The input to the buffer changes state, causing the simulator to schedule an event. When the event is processed, the simulator applies the new state to the input of the buffer and then evaluates the buffer to determine the effect of the altered input. The simulator then propagates the effects of the evaluated buffer to the buffer's output, which causes another event to be scheduled, perhaps with some delay. When QuickSim II processes this last event, the resulting value is typically fanned-out to other components.

Accurate simulation requires the ability to simulate state changes at the correct time and to model complex signal relationships. Because many simulation models have pin-to-pin propagation delays and pin rise and fall delays, the simulator must have a way to schedule signal state changes to occur in the future. QuickSim II schedules all events (current and future) using a "timing wheel" algorithm.



Devices that are used to change state, such as the Pullup component, do not schedule events, nor can they override the Xr state. However, if another even evaluates the net connected to such a device, the device will contend for the resultant value. Therefore, you must apply a "kickstarting" force (event) to the net where the device is connected to cause an evaluation to occur and propagate through the design.



- A slot holds all events occurring for that timestep
- Number of slots determined by memory and complexity of events.
- Mature events -- currently being evaluated
- Figure shows:
  - QuickSim II evaluating events from slot 3
  - Scheduling future events in slots 8, 13,....

### **Timing Wheel Concept**

A timing wheel initially consists of 1024 slots. QuickSim II can schedule events in any of the slots. Each slot holds all the events for that specific timestep of simulation time. The time of each timestep is set by the user, and defaults to 0.1 nsec. when the simulator invokes. If your models need finer resolution than this, you can set the resolution to 0.05 nsec. or faster. This requires memory to hold less of the simulation in the slots. You can also adjust the resolution to a coarser value, such as 1 nsec. to allow more events in memory.

Events in the slot for the current simulation time are considered *mature*. The simulator processes all mature events and then evaluates all the instances that those events affect. The result of evaluating each instance schedules new events in the future, based on timing delays. QuickSim II then sequentially processes the next slot(s) as simulation time advances.

The figure shows a conceptual timing wheel where QuickSim II is evaluating mature events in slot 3, and scheduling future events in slots 8, 13, and others.

If the delay of an event is greater than the amount of time in one "revolution" of the wheel (determined by the size of memory available), QuickSim II saves the event and schedules it later. If this happens frequently, it can decrease the simulator's performance.



For more information on the timing wheel concept, refer to the *QuickSim II User's Manual*.



If resulting events have 0 delay:

- Scheduled in next iteration for the current slot
- Events are mature, so the simulator performs another iteration for the current slot

Iteration limit (oscillation limit)--Allowable number of new events for current slot (iterations)



- Use the Set Iteration Limit command
  - o default is 1000
- Oscillation limit error
  - clears current events for oscillation loop
  - o does not clear current state in Monitor window
  - you must fix the oscillation, then issue stimulus and run again
## Iterations

QuickSim II reads the event list, processes the mature events and evaluates their logical effects on the circuit, and then schedules resulting events according to any associated delays. This three-step process is an *iteration*.

If any of the resulting events have a delay of 0, the simulator schedules them in a new event list in the current slot. Immediately, these events become mature, requiring the simulator to perform another iteration for the current slot (simulation time does not advance). The simulator repeatedly performs iterations until there are no more events in the current slot or an iteration limit is reached.

The table on the previous page shows what happens in the three inverter circuit when a "1" is forced on the input of the first inverter. The force is applied immediately to I1 and is evaluated. Since the output is delayed by 1ns, the change is scheduled in a future slot (2). Once this event matures, it is applied to the input of I2 and evaluated.

An *oscillation limit* occurs when required iterations exceed the maximum number allowed for the simulation. When this condition occurs, an Oscillation message warns you of the condition, the event queue is cleared for the current event slot, and the simulation comes to a halt. The default iterations allowed before and oscillation limit occurs is 1000, and is user settable with the **Setup > Kernel > Run Parameters** pulldown menu. You can also use the Set Iteration Limit command.

Most oscillation limit errors are caused by zero-delay gates with feedback. By default, unit delay will not incur oscillation limits, since all gates are simulated with a "unit" of delay. When timing mode is turned on, oscillations can occur. Add delays to all gates to remove this problem.



You can limit iterations with the Set Iteration Limit command described in the *Digital Simulators Reference Manual*.

# **Timing Accuracy**

Is a function of three things:

- Simulator's timing resolution (timestep)
  - defaults to 0.1 nsec set on invocation.
- Attributes of modeling methods; real-world timing (Kernel setup)
  - min/typ/max for rise/fall on I/O
  - min/typ/max for pin-to-pin timing from Technology files
  - Constraints Setup, hold, skew and minimum pulse width timing checks, and clock frequency constraints
  - Constraint violations that affect output states
- Special routines -- load-dependent delays (Technology files)
  - Process variation data
  - Pin loading and simulated physical effects

# **Timing Accuracy**

Timing accuracy in logic simulation is a function of three things: the simulator's basic unit of timing resolution, the attributes associated with the modeling methods, and special compensation routines.

The basic unit of time resolution for QuickSim II is the *timestep*, which is user definable and defaults to 0.1 nsec. when the simulator is invoked. QuickSim II uses timestep increments to model the passing of time during a simulation run. All simulation activity occurs within timestep boundaries. The smaller the timestep, the greater the timing resolution, and the longer the simulation run.

Modeling attributes help describe some of the real-world timing characteristics of component models. You develop modeling attributes during the schematic entry phase of the design process. Examples of the modeling attributes include:

- Minimum, typical, and maximum timing for rising and falling signals on inputs and outputs.
- Minimum, typical, and maximum pin-to-pin timing, which is defined in technology files.
- Setup, hold, skew and minimum pulse width timing checks, and clock frequency constraints.
- Timing constraint violations that affect output pin states.
- Timing as a function of pin loading and simulated physical effects.
- Transport delays for LM-Family models, and QuickPart Schematic models (but not for QuickPart Table models).

Compensation routines annotate the timing of a design with load-dependent delay estimates from physical layout data, process variation data, as well as temperature and voltage variances.

# **Setting Run Parameters**

#### Setup > Kernel > Run Parameters

Dum dimen	
Remove default run value	HDL assertion severity
Quiet	Warning
eration limit 1000	Error
DL array size 10	Failure

- Run time Limits "Run" to specific time.
- Quiet Stops simulation if no events scheduled
- Iteration limit Oscillation passes per timestep
- VHDL limits can be set

## **Setting Run Parameters**

You can customize the way you want a simulation run to behave in QuickSim II. This allows you flexibility and efficiency in performing simulation runs.

To change the way a simulation run is performed, choose **Setup** >**Kernel** > **Run Parameters.** The Setup Run Parameters dialog box appears, as shown on the previous page. The following describes how to setup run options:

- **Run time.** When you use the Run command without parameters, it will stop after this amount of time. Normally, the run does not stop. This setup is useful for many successive runs of the same length of time.
- **Remove default run value.** If an existing Run time has been specified, and you want to return to "no time" mode, click on this button.
- Quiet. Stops the simulation when an "all quiet" condition exists (no pending events are scheduled)
- Iteration limit. This specifies the number of times a zero-delay feedback loop will schedule successive iterations
- HDL array size and assertion severity. Allows you to set operation constraints for VHDL models. For more information on these parameters, refer to the Digital Simulators Reference Manual.

# **Using Breakpoints**

Pauses the simulation on predefined conditions

SETUP	Add Breakpoint
	On Expression VHDL Object
	Expression
	On change
	On occurrence 1 End of timestep
	Action list
	Stop simulation Delay actions
	Filter redundant events
	OK Reset Cancel Help

On Change -- Any state change, otherwise on True

On occurrence -- How many times must it occur

Action list -- QuickSim II command list to execute

Stop simulation -- paused, use resume to continue

Filter redundant events -- (default) does not respond to events creating the same state.

# **Using Breakpoints**

The breakpoint object allows you to specify the signal or condition (expression) that you are looking for, and have the simulation paused at that point.



There is a difference between stopping and pausing a simulation. When you stop the simulation run, any stop time is removed. You must then reissue the run to begin simulating again. When you pause a simulation, the original run length is remembered, and a resume command will continue to the original stop time.

To create a breakpoint, click on the **[Setup palette]** Add Breakpt palette button. The Add Breakpoint dialog box appears as shown on the previous page.

- **Expression**. Enter a signal name or expression of your choice. The signal state, or the expression evaluation determine if the break condition occurs.
- **On Change.** Allows the break to occur if the expression changes, because normally it only breaks if the expression goes from false to true.
- **On Occurrence.** Delay the break to the nth occurrence of the condition.
- End of Timestep. Delays the break until all timestep iterations are completed. This lets all zero-delay circuit activity stabilize before pausing the simulation.
- Action list. Allows AMPLEware to be executed when a break occurs.
- **Stop Simulation**. If you disable this button, the simulator recognizes the breakpoint, and performs any actions specified, but continues to run.
- **Filter redundant events.** Redundant events are those that produce the same state on a net or pin. For example, forcing a net to 1 that is already at the 1s state creates a redundant event. This button has these events filtered so that they don't cause a break condition to falsely occur.



For information about breakpoints, refer to the *SimView Common Simulation User's Manual*.

# **Using Actionpoints**

Allows a list of commands to run at a given point in the simulation, without stopping or pausing



To Add: Add > Actionpoint

Add Actionpoint		
Expression		
Action List		
On occurrence 1 I On Change		
Filter redundant events		
OK Reset Cancel Help		

Report: (Menu bar) > Report > Setup > Actionpoints

To Delete:

- Open the Actionpoints report window
- Select actionpoint entry
- Click on: DELETE

# **Using Actionpoints**

An actionpoint allows you to specify conditions which, when true, a set of functions (actions) are executed. An actionpoint behaves similar to a breakpoint that is executing an "Action List", except that the simulation will not be stopped.

Actionpoints are useful when you are debugging your design. You can use expressions that look for conditions in your circuit, and write any type of report, state, window, or result.

For example, you can set an actionpoint to write a report window when the message count equals 50 (one page full).

To create an actionpoint, choose the **Add** > **Actionpoint** menu item from the menu bar. The Add Action point dialog box appears, and allows you to enter the triggering expression, the occurrence of that expression, and whether the action occurs when the expression is true, or when it changes.



For information about actionpoints, refer to the *SimView Common Simulation User's Manual*.

# **Saving and Restoring Simulation States**

• Choose: <u>File > Save > State</u>

Save State			
Viewpoint			
Pathname	quicksim_state	Navigator	
Replace			
Query when Waveform DBs have edits pending			
OK Reset Cancel Help			

• Choose: <u>File > R</u>estore > <u>S</u>tate

Restore State		
Viewpoint		
Pathname quicksim_state	Navigator	
Restore state without confirmation		
OK Reset Cancel	Help	

- When viewpoint is selected, "Leafname" appears
- Simulator resolution must match saved state otherwise error is reported.

# **Saving and Restoring Simulation States**

QuickSim II allows you to save and restore the state of the simulator. This state object can also be saved to the viewpoint or to a non-viewpoint location.

To save the state of the simulator, choose:

#### (Menu bar) > <u>F</u>ile > <u>S</u>ave > <u>S</u>tate

The Save State dialog box appears. The dialog box provides a pathname entry area. You can use the navigator to find an alternate (non-viewpoint) location for the file. The default name for the file is *quicksim\_state*.

If you click on Viewpoint, pathname is replaced by leafname because you must place viewpoint objects in the viewpoint container.

If you have already saved a state under the same name, you must use the Replace option. This writes over the existing file. If you don't use the Replace option, you will get an error message.

To restore a state, choose:

#### $(Menu bar) > \underline{F}ile > \underline{R}estore > \underline{S}tate$

This action only restores the internal states of signals in the simulation. It does not restore window configurations, kernel checking configurations, or connected waveform databases. You must use the other Restore menu options to restore these environments.

# **Batch Simulations**

- What: Non-interactive simulations Scripts run the simulation in background You examine data later
- Why: Saves you time (you can do other things) Better utilization of QuickSim II (kernel) Simulation usually runs much faster

forcefiles **SimView** Stimulus **SimView** Waveform Setup **Databases** QuickSimII Results Waveform Database QuickSim Saved State Setup **SimView** 

How:

## **Batch Simulations**

Batch simulation refers to non-interactive simulation. To perform a batch simulation, therefore, you must gather together all of the interactive decisions and configurations in advance and provide this information to the simulator at invocation. You must also tell the simulator how much data to produce and in what form, prior to invocation. A single command can then configure the environment, perform the simulation run, and save the results for later viewing. Batch simulation can benefit you in the following ways:

- Save you time. While the simulation is running, you can be doing other things. For example, you can start a long simulation run at the end of the workday, and return the next morning to examine the results.
- Use QuickSim II kernel. QuickSim II is a combination of SimView and the QuickSim kernel, which actually performs the simulation. Much of interactive simulation consists only of using SimView to prepare or examine stimulus. These activities could be performed in SimView and allow QuickSim II to be used more efficiently for kernel simulations.
- **Batch simulation is faster.** Much of an interactive simulation overhead is spent updating displays. By invoking QuickSim II "displayless" and without stopping to update information (the kernel keeps the information until the end of the run), QuickSim II is most efficient.

The figure on the opposite page shows how you can take advantage of batch simulation performance by using SimView to perform certain tasks:

- 1. **Setup.** SimView can create, modify, and save waveform information to be used by QuickSim II. SimView can also convert other types of stimulus.
- 2. **Run.** QuickSim II is required to perform a simulation run. Setup and waveform information from SimView is connected, a run is performed, and results are saved.
- 3. **Examine.** SimView has full waveform examination capabilities. SimView cannot, however, load or view saved state information, nor use QuickSim II kernel setup objects.

# Setting Up Batch Simulations

Sample script:

// This is my\_dofile Dec 22, 1994
//
ADD LIsts clock clear b c d out
DOFile batch\_forces.do
LOAd WDb good\_wdb -Viewpoint
CONnect WDb good\_wdb 3000 -Absolute -Merge
RUN 190000
SAVe WDb my\_results results -Replace
\$set\_active\_window("List");
WRIte REport batch\_listfile 3000 190000 -Highlight
\$\$force\_exit()

# **Setting Up Batch Simulations**

Much of the setup for a batch simulation can be performed in SimView. SimView has the following capabilities:

- Full stimulus management capabilities. You can:
  - create stimulus
  - o edit waveforms and stimulus test files
  - create buses to combine common signals
- SimView environment setup, save and restore. You can use SimView to set up the simulation environment

You will also need to decide which kernel setup conditions you need in your simulation. You have two methods that can be used to set up the kernel:

- **Invocation Switches.** If you are setting up all objects in the design with the same kernel setup conditions, you can specify invocation options. Most of the kernel setup conditions correspond to an invocation switch.
- **Invocation Script.** If unique setup conditions are required, or setup conditions must change during the simulation, then you need to create an invocation script that executes after QuickSim II invokes. This can be of the following types:
  - A startup script. This script can be placed in your *mgc* directory or in the design, and is automatically run by QuickSim II when it invokes.
  - **Redirected input**. With this method, you create a file with the commands and functions. When you issue the quicksim shell command, you use the (<) symbol to apply the contents of the file as input to the application. An example of such a script is shown on the previous page.

This script can also be used to run the simulation, save the output, and exit when finished. Create "my\_dofile" by pasting the transcripted functions from a simulation run into a file. Delete unwanted functions or add other functions or commands. Place the file within the containment of the design so that it is managed with the design.

# **Running a Batch Simulation**

- Running with a setup object: shell> quicksim my\_design -nod -set my\_setup
- Running with a startup file: First, create the file in \$HOME/mgc/startup

shell> quicksim my\_design -nod

- Running with redirected input: shell> quicksim my\_design -nod < my\_dofile</li>
- Combinations of these modes can be used.

# **Running a Batch Simulation**

To run the simulator in batch mode using redirected input, you must invoke it from an operating system shell instead of from the Design Manager using one of two batch modes. When the dofile is completed, the simulator automatically returns control to the operating system shell. The following example uses "my\_dofile" as input to a QuickSim II simulation run:

```
$MGC_HOME/bin/quicksim my_design -NODisplay < my_dofile</pre>
```



Depending on the operating system used on your workstation, you may need to escape the "\$" symbol used in shell scripts. Once QuickSim II invokes, it handles dollar signs in dofiles properly.

It is a good idea to place files used within a batch simulation run within the containment of the design so that it is managed with the design configuration. In the above example, "my\_dofile" should be placed beneath "my\_design"

# Lab Preview

- Invoke QuickSim II on the OSC circuit
- Create and resolve a hazard condition
- Create and resolve an oscillation limit error
- Add a Probe to a net
- Change Rise and Fall delay properties

### Lab Preview

In the lab exercise for this module, you will:

- Invoke QuickSim II on the OSC circuit to be used to demonstrate simulation runtime concepts.
- Create zero-delay path within a feedback loop to cause a hazard condition. Resolve the hazard condition by changing timing delays.
- Create and resolve an oscillation limit error, and determine that scheduled events have been removed from the offending circuit path.
- Add a Probe to a net, thus creating a synonym for the net handle.
- Change Rise and Fall delay properties from the zero-delay values that are default for the gen\_lib components.

### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

#### **Procedure 1: Evaluations and Oscillations**

This procedure uses a simple circuit to show how QuickSim II evaluates design models. You will use a zero delay feedback loop to create an oscillation, then fix the oscillation by back annotating delays.

- 1. If necessary, log into your workstation and set your current directory to your *qsim\_n* directory.
- 2. Invoke QuickSim II on the OSC circuit using default shell invocation as shown in the following command.

```
shell> $MGC_HOME/bin/quicksim OSC
```

3. Open the root sheet for this design using the (open sheet) stroke.



This simple design will be used to illustrate several evaluation concepts. The pullup resistor sets the input of the first inverter to a 1r which should start the oscillation process. By modifying initialization values and timing properties, you can control how this circuit behaves.

4. Add probes to the circuit on each net as follows:



- a. To add the probe to the first net, select the net at the output of the first inverter (I1).
- b. Click on: **DBG GATES**



c. When the dialog box appears, enter the Probe name OUT1 and verify that the On "Selected Object" button is highlighted. **OK** the dialog box.

Add Probe			
Probe name OUT1			
On Selected Object Named Object Location			
Object type			
Any Instance Net Pin			
Replace existing probe or synonym			
OK Reset Cancel Help			

The probe object appears at the output of I1 just below the selected net.

d. Using this same procedure (a-c), add the Probes named OUT2 and OUT3 to the output nets of the other two inverters, respectively, as shown in the schematic view.

- 5. Create the Trace, List, and Monitor windows and add the new probe names to them as follows (read all steps and the note first):
  - a. Use the  $|| \downarrow |$  stroke to unselect everything in the circuit.
  - b. Using the  $\neg$  (Trace) stroke and the (List) stroke, create each of these windows, entering the probe names in the dialog boxes for each window.
  - c. Use the **DBG GATES** button and dialog box to add these

signals to the monitor window.



If you had selected each net, and added the selected objects to the window, you would have seen only the net handle (/N\$xx) instead of the new probe name. Try it to verify this.

6. Before running the simulator, visually check the circuit initialization.

Each net should be at Xr.

What do you think will happen at the input to the first inverter (I1) when you run the simulation?

7. Using any method you used in a previous lab, run the simulation for 10 nsec.

When the simulation stops, what are the current signal states?

OUT1 \_\_\_\_ OUT2 \_\_\_\_ OUT3 \_\_\_\_

The circuit did not oscillate because the values are X (unknown). This is because the pullup resistor did not override the Xr initialization value. Using the evaluation table "Multiple Driver Resolution" on page 6-6:

 $Xr + 1r = Xr \rightarrow INV \rightarrow Xs = X$ 

Xr processed through the inverter becomes Xs or just X, as shown in the List window. All nets in this design remain in this state. If instead you issued a 1s signal into the input of I1 (1s + Xr = 1s), the circuit would begin clocking. Let's give this circuit a "jump start."

- 8. Force a 1 into the input of the first inverter and run as follows:
  - a. Select the input net to the I1 inverter.
  - b. Click on: **STIMULUS** 1? - > 1 (**True**). This forces the selected net (OUT3) to a 1s.
  - c. Now run the simulation for another 10 nanoseconds.

Did the circuit begin clocking?

What is the period (time of one cycle) of the outputs?

Why?\_\_\_\_\_

- 9. Change the timing mode to Delay and run again as follows:
  - a. Choose: (Menu bar) > Setup > Kernel
  - b. When the dialog box appears, click on the "**Delay**" button.

Setup Analysis		
Timing mode Current Unit Delay Constraint		
Details of 'Current' timing mode Hidden Visible		
OK Reset Cancel		

If you make the details visible, you will see that "Typ" timing is set.

c. **OK** the dialog box.

Before you run the simulation again, what are the delays for each of the inverters?

What will happen in the circuit?

d. Run for another 10 nsec. to verify your assumptions.

An oscillation limit occurs at the first event (time = 20.0 nsec.) due to zero delays for the rise and fall timing for each inverter. You will fix this problem in the next step.

e. Close the Info Messages window.

- 10. Back annotate the Rise and Fall properties as follows (schematic view is active):
  - a. Click on: DESIGN CHG



- b. Place the crosshairs on the top (Rise) property for I1 and click.
- c. When the "Change Property: rise" dialog box appears, enter the following and then **OK** the dialog box:

Change Property: RISE		
Value 1	Property Type	
BA Nameqsim_n/OSC/default	<ul> <li>Number</li> <li>Expression</li> <li>Triplet</li> </ul>	
OK Reset Cancel		

The Rise property for I1 should now be a "1" and in red, indicating it is a back annotation. If the property doesn't change, and you get an "invalid...ba\_name" message, do the procedure again.

d. Now, use the procedure (a-c) to change the Fall property value to **1**. Your schematic view should now look like the following:



What will happen in the circuit on the next run.

e. Now run the simulation for 10 nsec. to confirm your assumptions.

The circuit did not toggle. This is because the oscillation condition

cleared all pending events for the current time. Since these were the

**Note** only events scheduled, the nets stay in the same state. To fix this condition, you must add an event (force a signal state).

- 11. Schedule an event, and run again as follows:
  - a. Select the input net to I1 and use STIMULUS force this signal to a 1. STIMULUS O =
  - b. Now run the simulation for 10 nsec.
- 12. Set the Fall property for I1 back to 0, turn on Hazard checking and Run again by doing the following steps:
  - a. Using the method you used to set the Rise and Fall properties to 1, set the Fall property for the first inverter (I1) back to a value of 0. Note that a triplet value is used.
  - b. Then choose: (Menu bar) > Setup > Kernel
  - c. When the dialog box appears, click on **Visible**.
  - d. Turn on hazard checking (Hazard check).
  - e. **OK** the dialog box.
  - f. Now run the simulation for 10 nsec.

The Simulation Messages window reports hazards that have occurred on each instance for each event during the simulation. That's because hazards occur when an instance changes state more than once per time step. Each signal transition makes two passes of the circuit feedback loop before moving to the next event.

- 13. Close the Simulation Messages window.
- 14. Finally, set the Fall property for I1 back to 1 and run for 10 nsec.

This fixes the hazard problem because there is now a delay in both Rise and Fall paths through the circuit.

In the next procedure, you will examine the effects of simulator resolution on timing values during a simulation run. Do not exit QuickSim II, since you will use the OSC circuit in its current state.

#### **Procedure 2: Resolution and timing**

In the following procedure, you will adjust the simulator resolution to determine its effect on the timing values used in a simulation

- 1. Using the OSC circuit from the previous lab procedure, do the following:
  - a. Set the Fall property value for I1 to 0.1 nsec (the current resolution of the simulator) using any method.
  - b. Run the simulation for 10 nsec.

Note the new duty cycle for the pulses that are generated. The period is 1.1 nsec with a pulse width of 0.1 nsec.

- 2. Adjust the Fall property value for I1 to half of the simulator resolution and run again as follows:
  - a. Using any method, set the I1 Fall property to 0.05 nsec.
  - b. Run for 10 nsec

What is the period \_\_\_\_\_\_ and pulse width \_\_\_\_\_ now.

3. Set the fall property to 0.04 nsec and run for 10 nsec.

Note that the pulses have disappeared altogether and the simulator reports that hazards have occurred. The 0.04 nsec delay is too small to be recognized by the simulator.

4. Save the design changes that you have made to create a persistent viewpoint as follows:



- 5. Save the current state of the simulator as follows:
  - a. Choose: (Menu bar) > File > Save > State
  - b. Fill out the dialog box as follows:

1 Click on				
	Save State			
	Viewpoint	2. Enter		
	Leafname	osc_state		
	Replace	3. Click off		
	_ Query whe	n Waveform DBs have edits pending.		
4. Clic	к ок	Reset Cancel Help		

- 6. Exit QuickSim II saving all setup conditions to a viewpoint object named "osc\_setup".
- 7. Invoke QuickSim II with 0.01 nsec resolution, restoring the state, by entering the following shell command:



QuickSim II reports an error message:

ERROR: Cannot specify kernel preload switch(s) when specifying '-restore' Type '\$MGC\_HOME/bin/quicksim -help' for help.

This is because the state you are restoring contains the simulator resolution and can't be changed with the -ts switch. In other words, -time\_scale and -restore\_state can't be used in the same command.

8. Again, invoke QuickSim II with 0.01 nsec resolution, by entering the following shell command:

```
your_path/training/qsim_n
shell> $MGC_HOME/bin/quicksim OSC -ts 0.01
-setup OSC/default/osc_setup
```

Notice that this invocation gives you the same message. This is because the setup file contains the simulator resolution, which conflicts with the value of the -ts switch.

- 9. Invoke on OSC once more, and restore a state, as follows:
  - a. Invoke QuickSim II once more using the command below:

your_path/training/qsim_n				
shell>	\$MGC_HOME/bin/quicksim	OSC -ts	0.01 -tim	typ

b. Choose: (Menu bar) > File > Restore > State and fill out the dialog box as follows:



- c. You get the message "Restore state operation failed." This is because the saved state simulator resolution (0.1nsec) does not match the current simulator resolution (0.01nsec). To restore this state, you would have to reinvoke QuickSim II with a resolution of 0.1nsec.
- 10. Perform a final run as follows:
  - a. Open the schematic view window.
  - b. Select the net input to I1, then trace and list it.
  - c. Force the selected input to a 1 using the "Force to State" palette button.
  - d. Run for 10 nsec.
  - e. Examine the trace for the input net to I1.

The trace shows that the Fall property value (0.04) now resolves.

11. Exit QuickSim II without saving anything.



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 8 Solutions" on page A-15.

Then continue on to the next module.

## **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 8 Solutions" on page A-15.

- 1. What three categories determine the timing accuracy in a digital simulation?
  - 1.

     2.

     3.
- 2. What is the difference between an iteration and an event?

\_\_\_\_\_

- What conditions can occur when a signal enters a zero-delay feedback loop?
   a. \_\_\_\_\_\_
   b.
- 4. List the two modes of resetting the simulation state and describe the difference. 1. \_\_\_\_\_ 2. \_\_\_\_\_ 2. \_\_\_\_\_
- 5. How does an actionpoint differ from a breakpoint?
  - 1.\_\_\_\_\_ 2.\_\_\_\_
- 6. What are the steps you use to create a batch simulation?
  - 1.

     2.

     3.

4.\_\_\_\_\_

- 7. Which menu item do you use to view the current breakpoints.
- 8. Describe the function of the breakpoint action list. \_\_\_\_\_
- 9. When does a breakpoint occur if you set "On Occurrence" = 0x10?

## **Module Summary**

The QuickSim II kernel uses a timing wheel algorithm to schedule simulation events and keep track of simulation states. An event is when a state change occurs. A mature event is one that is in the current time slot. An iteration is a portion of a time step where all models are evaluated only once. Iteration limits prevent the simulator from "hanging" on zero-delay feedback loops.

The Add Breakpoint command allows setting up conditions that cause the simulation to pause so that you can examine data. The "On occurrence" option delays the break to the n<sup>th</sup> occurrence of the condition. You can also choose to pause at the end of the timestep, or immediately. The "Action" field specifies a function to execute when the break occurs.

Batch simulations are useful for using workstation CPU time effectively. Simulation stimulus can be compiled and kernel conditions set up in configuration files. The simulation can be run in displayless mode, logging data to be viewed at a later time.

Several new switches have been provided to the quicksim command in V8.

- 1. **-Display**. Turns off display mode so kernel does not send information to the frontend process for display.
- 2. **-DeFaults setup\_name**. Uses a setup file to set initial conditions.
- 3. **-REStore** *save\_state\_obj*. A state file to use at the beginning of simulation.
- 4. Other switch options allow presetting all of the kernel checks.

Many batch simulation modes and options are available. You can choose the methods that best suit your digital analysis environment.

# Module 9 Examining Results and Waveforms

Overview	9-2	
Lesson	9-3	
Results Analysis in the Design Flow	9-4	
Examining Results (SimView)	9-6	
Palettes Provide Access	9-8	
The Trace Window	9-10	
Trace Window Cursors	9-12	
The List Window	9-14	
The Monitor Window	9-16	
Using Monitor Flags	9-18	
Waveform Edge Deltas	9-20	
Viewing a Specific Time	9-22	
Bus Structures	9-24	
Defining Buses (setup)	9-26	
Assertions	9-28	
Assertion Tests	9-30	
Converting Waveforms to Assertions	9-32	
Lab Preview	9-34	
Lab Exercise	9-36	
Procedure 1: Analysis Using SimView	9-36	
Procedure 2: Analysis Using Assertion Tests	9-45	
Test Your Knowledge	9-50	
Module Summary	9-51	

#### **Overview**


### Lesson

On completion of this module, you be able to:

- Edit the events in a waveform.
- Analyze the events within a waveform using edge deltas and trace cursors.
- Create a waveform pattern for a bus using the pattern generator.
- Convert waveforms to assertions for use in waveform testing
- Report and view assertion tests and failures



You should allow approximately 1 hour and 20 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

# **Results Analysis in the Design Flow**



- Graphically analyze results waveform data using palette icons
- Optionally, save interim stimulus and results data
- Modify stimulus waveform data (optionally modify property values as back annotations)
- Reset State and rerun simulation with new stimulus

## **Results Analysis in the Design Flow**

Once you've applied stimulus to the design and run the simulator, results waveform data is created. The results data should be analyzed, modified, and saved as follows:

- 1. Check the actual data created by the simulation run against the expected data. Optionally, you can invoke SimView to graphically analyze the results waveforms and to perform any stimulus waveform modifications that are deemed necessary. Save any stimulus that you create or modify. If the actual data created by the simulation run matches the expected data, then you're design is ready for the next step - either fault analysis or layout.
- 2. Optionally, save the interim stimulus and results data.
- 3. If the actual and expected data do not match, you need to perform one of the following and then rerun the simulator:
  - Modify design property values to correct design errors or to perform "what if" analysis. Note that the simulator places all property changes that you make interactively in a back annotation object.
  - Modify existing stimulus or create new stimulus. Although a viewpoint is not required, it's good design practice to use one.
  - Change your expectations (expected results)
- 4. Reset the state to time zero and rerun the simulator. Simulate the design (with a design viewpoint attached) using various combinations of checking. When the simulation is complete, repeat steps 2 through 4 until the design produces the desired simulation results. Note that you can perform this step in simulation batch mode to speed the overall run time.

This module teaches you various methods for determining whether the actual results match the expected results.





- Palette lcons for ease of use
- Trace, List, Monitor, Monitor Flag, Report
- Trace Cursors, View specific time, View Area
- Report Edge Deltas, Timing, Changes

## **Examining Results**

SimView is the stand-alone version of the SimView/UI that is connected to the front end (user interface) of QuickSim II. It has all the viewing, expression, waveform editing, and saving capabilities as does QuickSim II.

Assuming that you have saved all of the information you want to examine from a QuickSim II batch run, you can invoke SimView or QuickSim II on your design and examine the results. Here are the things you can do in either SimView or QuickSim II:

- View the design sheet.
- Create Trace, List, and Monitor windows to examine signals.
- Load waveforms for viewing, and save waveforms in forcefile and logfile form.
- Edit waveform data using the waveform editor.
- Create expressions to compare and contrast data.
- Set up the SimView environment, and save it to a file.

Here are a few things you can do only in QuickSim II:

- Run a simulation.
- View VHDL source in the VHDL window.
- Restore or examine save state information.
- Restore or examine QuickSim II kernel setup information.

# **Palettes Provide Access**

 Graphically analyze and modify waveform data using palette icons (QuickSim II Palettes)







## **Palettes Provide Access**

Use the palettes to modify and analyze waveforms. More information about each palette icon is in the **Help > On Palettes > Palette Descriptions** menu items.

The **Stimulus** palette (not shown) provides icons that allow you to add and delete individual forces as well as clocks.

The **Waveform Editor** palette allows you to graphically manipulate waveform events.

- Add "Force Target" waveforms to Trace window for editing
- Graphically insert, copy, invert, move, or delete events

The **Debug Gates** palette provides several icons that assist you in analyzing a waveform.

- Create and move trace cursors that indicate the time and state of each waveform in the Trace window
- Report edge deltas to assist you in determining delays and timing
- Add Monitor windows and Monitor Flags to view current state values
- Locate and backtrace to their source unknown X values (see Module 11)

The **Analyze** palette provides icons that allow you to perform waveform testing (expected results waveform(s) vs. actual results waveforms)

- Add and delete assertions (expected results)
- Convert waveforms to assertions for use in waveform testing
- Report and view assertion tests and failures

# **The Trace Window**



- Change attributes All Trace windows: (Menu Bar) Setup > Window Attributes > Trace Defaults
- Change attributes Active Trace window: (Popup Menu) Setup > Window

### The Trace Window

A Trace window is a graphical display area for digital logic waveforms. In this window, you provide pin, net, bus, or waveform names and QuickSim II graphically traces either the default waveform for a pin, net, or bus or traces the specified waveform. The waveform traces appear similar to that of an oscilloscope or logic analyzer.

Logic levels are shown in the Trace window relative to the tic marks (+):

- Through the tic mark reflects an unknown (X) value
- Above the tic mark reflects a high (1) value
- Below the tic mark reflects a low (0) value

The drive strengths are shown in the Trace window using both line types and colors as shown in the figure.

A gray vertical line in the Trace window indicates the current simulation time. Result (output) traces stop at this line. Traces of stimulus signals can extend beyond this line. Times beyond the current simulation time are shown in italic.

When a Trace window is created, the default window name is "Trace". Multiple Trace windows can be created. Additional Trace windows are named "Trace#2", "Trace#3", and so forth. To create additional Trace windows, choose the (**Menu bar**) > **Setup** > **Open New Window** > **Trace** menu item.

To remove a signal from the Trace window, first select the trace(s), then click on the **Delete** palette button.

To move a signal within the Trace window, first select the trace. Then use the stroke. A prompt bar appears and the mouse pointer changes to a crosshair locator. Position the crosshairs to the new location, then click the Select mouse button.



For more information on using the Trace window, refer to "Trace Windows" in the *SimView Common Simulation Reference Manual*.

## **Trace Window Cursors**

- Shows state of each traced net at a given time
- Displays time at cursor position and cursor name

/COUNT(11:0) + 1FE 1FE 1FF + + Trace		$\Box \Delta$				
/START + + 1 + + + + <b>Open</b>	$\triangleright$					
$\underline{A}$	$\triangleright$					
	$\triangleright$					
$/CLR + + 1r^+ + + + Force$	$\triangleright$					
ANALOG_OUT + + Xi + + + Edit						
/OSC + + 0 + + + <b>Select</b>						
<u>Unselect</u>	$\triangleright$					
Cursor1 Time(ns) Setup	$ \ge $	$\bigtriangledown$				
View Cureere						
Cursors Cursors Label Interval						
To add a cursor, click on: DBG GATES						
ADD CU Cursor Name Location choices: Time 0 Click 🗏 OK	Са	ncel				

#### To move a cursor:

 Move the mouse pointer over the cursor (pointer becomes
 Click on the cursor and drag to new location.

### **Trace Window Cursors**

The Trace window supports multiple cursors. Each cursor provides all of the signal states at that moment in time. Simulation performance is not degraded by multiple cursors.

To create a cursor:

1. Click on: [Debug Gates] Add Cursor palette button.

A prompt bar appears requesting you to supply the time and the name for the cursor.

- 2. Enter the name you want to assign the new cursor.
- 3. Either specify the time and **OK** the prompt bar, or click on "+" followed by the location on the Trace window where you want the cursor placed.

The cursor appears in the active Trace window.

A cursor can be relocated within a Trace window in several different ways depending on accuracy in which you want the placement. The least accurate method but the simplest is to click on the cursor and dragging it to a new location. The more accurate methods are available through the [Debug Gates] Move a Cursor palette icon.

The Slide/Snap Cursor prompt bar is displayed providing you the following types of Trace Cursor movement:

Exact slide - similar to the click and drag method. Snap to nearest edge - places cursor on the nearest event. Snap to next edge - places cursor on the next event. Snap to prev edge - places cursor on the previous event. Snap to tic - places cursor on the nearest tic mark.



For more information on Trace window cursors, refer to "Using Cursors" in the *SimView Common Simulation User's Manual*.



# **The List Window**

Use List window attributes to control when new lines are added:

On Change = add a line every time a listed object's value changes (default)

List Period and No Change = add a line at regular intervals (determined by the list period value)

• Change attributes - All List windows:

Setup > Window Attributes > List Defaults

• Change attributes - Active List window:

(Popup Menu) Setup > Window

## **The List Window**

The List window is a textual view of signal or waveform information. Stimulus can also be previewed in text form before running a simulation.

The List window presents signal changes in bold (green) text, so you can easily identify where these transition occur. If you specify "On Change" when you add signals to the List window, a change to any signal will list the time and states of all signals. Otherwise, signal states are only listed at regular intervals as determined by the "List Period."

The default window name is "List". Multiple List windows can be created. Additional List windows are named "List#2", "List#3", and so forth. To create additional List windows, choose the (**Menu bar**) > **Setup** > **Open New Window** > **List** menu item.

To remove a signal from the List window, first select the list item(s), then click on the **Delete** palette button.

To move a signal within the List window, first select the list item. Then use the  $\searrow$  stroke. A prompt bar appears and the mouse pointer changes to a crosshair locator. Position the crosshairs to the new location, then click the Select mouse button.

To select an item in the list, place the pointer in the list selection box (just above the signal name) or on the signal name and click the Select mouse button. A shaded selection area appears.



For more information on using the List window, refer to "List Windows" in the *SimView Common Simulation Reference Manual*.

## **The Monitor Window**

### • Shows the Current State (current simulation time)

		Monitor			
Time(ns)	/count(5)	/count(4)	/count(3)	/count(2)	$\Delta$
6.5000.0	<u><u>1</u></u>	0	0	<u>0</u>	$\overline{}$



	Add Monitors			
On	Selected Signals Named signals			
	Signal name			
Buse	es Default Combine Flatten	lay radix Default Hex Octal Binary Decimal Binary Decimal		
	Use a mapping formatter			
		Select		
	OK Reset Cance	el Help		

### **The Monitor Window**

The Monitor window provides a snapshot of the current simulation states and time in text form. Unlike the List and Trace windows, no signal history is available in this window.

You can choose the "Type" of monitor you wish. If you choose the "Window" type of monitor, the signal is added to the Monitor window as shown in the figure. If you choose the "Flag" type of monitor, a state flag is placed in the schematic view window next to the net that you have selected. Flag monitors are discussed in the next lesson.

The default location for the Monitor window is at the top of the session area. The default window name is "Monitor". Multiple Monitor windows can be created and are named "Monitor#2", "Monitor#3", and so forth. To create additional Monitor windows, choose the (Menu bar) > Setup > Open New Window > Monitor menu item.

To add signals to a Monitor window click on [Debug Gates] Monitor Window.

If no signal is selected, a dialog box as shown in the figure appears that allows you to name the signals that you want to add.

To remove a signal from the Monitor window, first select the object(s), then click on the **Delete** palette button.

To move a signal within the Monitor window, first select the monitor item. Then use the  $\$  stroke. A prompt bar appears and the mouse pointer changes to a crosshair locator. Position the crosshairs to the new location, then click the Select mouse button. You can move or delete the time area also.



For more information on using the Monitor window, refer to "Monitor Windows" in the *SimView Common Simulation Reference Manual*.

# **Using Monitor Flags**

• To Add: Select nets and click on: DBG GATES



• Displays the current state of the net/pin.



 Can display state information for a specific time by interactively changing time domain: (Schematic Popup) Flag Monitors > Change Domain



## **Using Monitor Flags**

Flag Monitors are graphical objects that contain the current state of the pin or net., and are placed on the schematic sheet connected to the signal. Optionally, you can change the domain reflected in the Flag Monitors by choosing the (schematic view) Flag Monitors > Change Domain popup menu item. By doing so you can look at the values associated with any time value prior to the current simulation time.

To create a Flag Monitor you first select the nets or pins whose activity you want to monitor. Then click on the **[Debug Gates] Monitor Flag** palette icon. Flag Monitors are added to each of the selected nets.

Two types of flags are created depending upon the object chosen. If you select a pin, a pin flag is used. It uses colors similar to pin default colors (purple on a standard workstation) and the flag is placed above the pin. If you select a net, the flag uses net default colors (orange) and the flag is placed below a net vertex.

The flags are persistent, that is, they can be added to a schematic view even if the view is not present. When you finally open a view at that level, you will see the flags. If you close the schematic view that contains Flag Monitors, and then reopen the same view, the flags will appear as before.

To remove a Flag Monitor from the schematic view window, first select the flag, then click on the **Delete** palette button. The graphical flag is removed.

To dynamically move along the time domain to view the associated state values within a Flag Monitor, choose the (Schematic) Flag Monitor > Change Domain popup menu item. The Flag Monitor Domain dialog box is displayed and will remain displayed until you click on the Close button.

From within the Flag Monitor Domain dialog box you can navigate the time domain in several different ways: Move up or down the time domain by what ever increment you specify, go directly to a New time, or Reset back to the current simulation time (Sim domain).

Flag Monitors cannot be moved.

# **Waveform Edge Deltas**

### Measure time between Trace signal transitions

Click	con: s	TIMULU	S _		-							
				Tr	ace							
	/CLR	+	+	+	+	+	+	+	+	+	+	$\left  \begin{array}{c} \\ \\ \\ \end{array} \right $
ANA	LOG_OUT	4					+	+	+	+	+	
	/OSC	+	+ 6	6424n	s <sub>+</sub>	+	+		+	+	+	
	610	0.00	6200.0	0 6	300. 7	0 Cime(1	6400 ns)	.0	6500	.0	6600	$\bigtriangledown$
	/											
	Nearest Ec	lge St	ate	Time								
		EAR TH	e firs	T EDO	GE L	ocatio	on 1	÷	ок с	Cance	el	

### **Click Select mouse button near transition:**

#### Message area:

**i** Edge #1: results@@/i\$10/osc:pin = = 0 at time 64212 (ns)

CLICK NEAR/TOWARDS THE SECOND EDGE Location 1 🗄 OK Cancel

#### **Click Select mouse button near second transition:**

#### Message area:

**i** Edge #1: (ditto), Edge # 2: results@@/i\$7/analog\_out:pin = = 0

at time 64354 (ns), DELTA = = 142 (ns)

### **Waveform Edge Deltas**

Analysis of timing and functional information from text windows becomes increasingly difficult as the complexity of the design increases. It is much easier to use the graphical information presented in the Trace window. To help you with this analysis, the Waveform Deltas function allows you to specify a beginning and ending signal transition and determine the time between them.

To issue this function, you click on the **[Waveform Editor palette] > Deltas** icon. This brings up the graphical deltas cursor within the active Trace window. As you move this cursor, a vertical position flag displays the cursor time and the state of the signal that the flags is positioned over. The figure on the previous page shows this.

You are also presented a prompt bar, shown in the second figure, that allows you to grab the time of the nearest transition of the flagged signal. Use the Select mouse button to execute this operation. When you execute this prompt bar, the message area gives you information about the nearest transition; specifically, edge number (1 or 2), signal name, signal state, and time of transition.

Next, a second prompt bar appears, which asks you to grab the time of a second edge, the same way you did the first edge. Position the cursor both horizontally and vertically to the second transition. This transition can be on *any* signal in the Trace window. This time when you select the location, the message area adds the information about the second edge, including the computed time difference between the two edges.

You are then returned to the first prompt bar again. This process continues until you choose Cancel on one of the prompt bars.



For more information about calculating waveform deltas, refer to the *SimView Common Simulation Reference Manual*.

# Viewing a Specific Time

Use Trace and List window popup menus

• View Time by Selection

Useful for troubleshooting messages

- Select a time (maybe in a report window)
- In Trace or List window, choose:

(Popup) > View > Time > Selected

• View Specific Time

Quick method of scrolling a Trace or List window to a specific time

• In Trace or List window, choose:

(Popup) > View > Time > Specified:



## **Viewing a Specific Time**

A specific time can be viewed in the Trace or List window.

You can view a specific time in a window by selection; that is, select a time in one window and view that time in another. This is a very useful troubleshooting technique for examining signals and states at a reported error or condition time.

To view a selected time:

- 1. Select the time in a report window.
- 2. Move the pointer to the window that you want to adjust to that time.
- 3. Choose the (**popup**) > **View** > **Time** > **Selected** popup menu item. The window will adjust the view to the selected time. For example, the time will appear at the right side of the Trace window, or at the bottom of the List window.

To view a specified time:

- 1. Choose the (**popup**) > View > Time > Specified: menu item. A prompt bar appears as shown.
- 2. Enter the time, and use the stepper to specify whether that time is relative or absolute.
- 3. **OK** the prompt bar. The active window displays information at the specified time.



For more information on viewing by time, refer to the *SimView Common Simulation User's Manual*.

### **Bus Structures**

Bus -- group of wires grouped as a single name.

- Assigned in Design Architect
- Can be defined in QuickSim II or SimView

**Bus Naming Conventions:** 

- Uses subscripts to identify size of the bus ADDR(15:0) DATA(31:0)
- Subscripts also provide bit mapping: ADDR(15:0) == 000F ADDR(0:15) == 000F FAXPORT(7:0) == DATA(31,6:0)
- Trace or List "combined" or as individual wires.

## **Bus Structures**

A bus is a group of wires that are referred to by a single name. It is a convenient way of referencing a group of related signals. Common buses in digital logic design are Address buses, Data buses, Port connections, and Control lines. Buses can be created in the following ways:

- Assigned in Design Architect. By creating nets and assigning a net name in bus designation (see below), you can create a bus.
- Defined in QuickSim II or SimView. You can associate multiple wires as a single bus by using the Create Bus command or operations. The next pages shows you how to do this.

Buses are named using a text string followed in parenthesis by a numeric range of subscripts. The following shows how these subscripts are used:

- The colon (:) is used to identify a range of values in identifying the size of a bus. For example, ADDR(15:0) represents a 16-bit address bus named ADDR. The bus DATA(31:0) is a 32-bit bus. If you wanted to trace the most significant bit of the DATA bus, you would identify it as DATA(31).
- Subscripts also provide bit mapping when you are using bus nomenclature in expressions. For example, ADDR(15:0) == 000F is not the same as ADDR(0:15) == 000F, since, in the first case, bits 3-0 of the bus are "1" and in the second case, bits 15-12 are ones.

The following example shows how you can map bits in one bus to those in a new bus for display purposes. FAXPORT(7:0) == DATA(31,0:6) maps DATA bit 31 to FAXPORT bit 7 and DATA bits 6-0 to FAXPORT bits 0-6. The DATA bit 0 maps to FAXPORT bit 6.

• The Trace and List options allow you to either display the "combined" bus as a single trace or value, or as individual wires (flattened). This can save you time when you place signals in these windows.

# **Defining Buses (setup)**

	Add Bus	
Bus name	Replace	Combine
Source Nets Selected Nets:	>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>	New Bus Bus Elements
dout2 dout3	Add to Top -> Add to End -> Only use SELECTED:	
7	Add to Top -> Add to End ->	7
	**** HOW TO USE **** Choose from the list of	Total count: 0 Edit Operations
Other Nets	Selected nets and/or by specifying Other nets. Reorder the new bus by using the Edit operations.	The Edit operations work on the bus elements in the list box above.

# **Defining Buses**

A *bus* in SimView is a group of signals that are grouped under a common name. If signals are viewed together, or behave as a bus, it is advantageous to handle them as a bus.

Buses can be defined using the Design Viewing and Analysis (DVAS) commands. SimView has a special **Add Bus** command that allows you to assign viewing characteristics to the bus.

To create a bus in SimView (or SimView/UI operating with a simulator):

- 1. Select the nets that you want grouped as a bus.
- 2. Click on the [Setup] Define New Bus palette icon.
- 3. The Define a New Bus dialog box appears. Assign the bus name in the entry area. Add the nets to the bus elements list to create the bus.

The combine option allows you to view the signals as if the bus were created in Design Architect. This is useful for signals that are bit related, such as outputs of registers, data masks, addresses, and data channels. The defined bus name lists, traces, and monitors as a single signal.

If you do not select the combine option, signals can be selected, added to windows, and removed in groups. But if you add the group name to a Trace, List or Monitor window, each of the signals is entered separately.

4. **OK** the dialog box.



For information about defining buses, refer to the *Design Viewing and Analysis Support Manual*.



- Assertions: special waveform data that represent Expected Results and that reside in asserts WDB
- Assertion Tests: expressions that represent the compare equation and that are automatically created when you issue an assertion
- Issue Assertions with:
  - Assert command
  - o Assertfile
  - Click on: ANALYZE



Single Assertion... Multiple Assertions... From File...

### Assertions

Assertions are a special form of waveform data that represent the events that you expect to occur on a given pin, net, or bus. The assertions reside in a special, predefined waveform database, called asserts.

Assertion Tests are expressions that compare the actual results (kept in the results waveform database) to the expected results (created in the asserts waveform database). When you create an assertion an assertion test is automatically created. The assertion test contains the assertion waveform name as well as the associated results waveform name. In this way whenever a simulation run is performed the assertion test is evaluated and pass/fail results are available for analysis in either the Trace or List window.

You can create assertions for the asserts waveform database in several ways:

- Select a net and click on the Add palette button in the Analyze palette, and choose one of the three menu items (see previous page)
- Issue one or more Assert commands (very similar to the Force command)
- Create an assertfile (an ASCII file similar to a forcefile)

## **Assertion Tests**

### (Menu Bar) Report > Waveforms - asserts and results



## **Assertion Tests**

You create assertion waveforms that represent the expected results for a specified pin, net, or bus. The assertion waveforms are placed in the "asserts" waveform database. You report the assertion waveforms in the asserts report window by choosing the Report > Waveforms pulldown menu item. You can also trace and list the assertion waveforms just as you would any other waveform.

The assertion tests are expressions that represent the comparison of the expected and actual results data. You report a list of the existing assertion tests by clicking on the [Analyze] Report palette icon.

When you run a simulation, the actual results data is kept in the "results" waveform database. At the same time the assertion tests are evaluated producing a pass/fail output that you can view in the Trace or List window. To view the evaluated results of the assertion tests, click on the [Analyze] View Failures palette icon.

# **Converting Waveforms to Assertions**



### **Converting Waveforms to Assertions**

The figure illustrates a simple two-phase debug process. In this process it is assumed that the final expected results is known and therefore, assertions can be created to test against the actual results.

While performing the debug process Debug 1, the forces are modified and the simulation is run until the actual results pass the assertion tests against the expected results. At this time the forces to Debug 1 are known to be good.

With the Debug 1 forces tested as good they are now considered the expected results for the debug process Debug 2. However, rather than creating the assertions for Debug 2 you can convert the forces waveform database to assertions by clicking on the **Cvt Wfs to Asserts** button in the Analyze palette.

## Lab Preview

- Invoke SimView without a design
- Load results waveform data and examine
- Load the design
- Create a user-defined bus, stimulus pattern for the bus, and assertions from the stimulus
- Exit SimView and invoke QuickSim II
- Load the stimulus used in the previous run
- Load existing assertions and run the simulator
- Examine the assertion test failures
- Modify the stimulus and rerun the simulator
- Exit QuickSim II

### Lab Preview

In the lab exercise for this module, you will:

- 1. Invoke SimView without a design.
- 2. Load and examine the results waveform data from the previous module.
- 3. Load the add\_det design upon finding errors in the results.
- 4. Create a user-defined bus, apply a stimulus pattern to the bus, and create assertions to be used in the next QuickSim II simulation run.
- 5. Exit SimView.
- 6. Invoke QuickSim II on the add\_det design.
- 7. Load the stimulus that was used in the batch run you performed in Module 8.
- 8. Load the assertions created during the SimView session. This will automatically create the assertion tests.
- 9. Run the simulator to exercise the assertion tests.
- 10. Examine the assertion test failures.
- 11. Modify the stimulus to correct the test failures.
- 12. Run the simulator again to verify that the assertion tests pass.
- 13. Exit QuickSim II

### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

In this lab exercise, you will analyze the results data created by the run in the previous module. The initial analysis and assertion test creation will be performed using SimView. This is to illustrate how SimView can be used independent of QuickSim II to manipulate waveform data. These same procedures could also be performed from within a QuickSim II session.

Later in this lab exercise, you will perform assertion runs and tests using QuickSim II.

### **Procedure 1: Analysis Using SimView**

- 1. If necessary, log into your workstation and set your current directory to your *qsim\_n* directory.
- 2. Invoke SimView without a design by issuing the following command:



The SimView session window appears.

Notice that the **[Setup] Open Sheet** palette icon is shaded because there is no design loaded for which to open a sheet.

- 3. Load the results data from the previous lab into a results waveform database by performing the following:
  - a. Click on the **STIMULUS**

palette icon:

The Load Waveform DB dialog box is displayed.

- b. Use the **Navigator...** to select the **results** waveform database file under the add\_det design.
- c. **OK** the Navigator dialog box; the path to the *results*\_ file is displayed in the Pathname entry field.
- d. Fill out the remainder of the Load Waveform DB dialog box as shown:

Load Waveform DB				
Viewpoint	(Not available until design is loaded)			
Pathname re	esultsSvdm_svdb.attr Navigator			
Load into t	the 'forces' WDB < 1. Do not Selec			
WDB name	results			
<b> Repeat</b> (U	Jsed with Logfiles only) 2. Enter name			
Should the WDE WDB may be co	B be connected now for use as stimulus? If not, the onnected later using the "Connect WDB" command <b>/aveform DB immediately</b>			
	3. Click on			
C	Cancel Help			

The "results" waveform report window appears. The report window lists the waveforms just loaded into memory.

4. Add the FULL, PARITY, and ACCESS results waveforms to a Trace window by selecting them in the results report window and then using the Trace stroke. Hint: select the ACCESS bits in order (15-0).

As you can see, the ACCESS bus has been flattened into its individual bits. This is due to a limitation in the waveform database storage format. All buses are flattened when saved and cannot be combined when reloaded. This limitation only applies to saved waveform databases that are reloaded. Bus information that is created in program memory can be flattened and combined at will.

- 5. Add the selected results to the List window by using the List stroke. You may find it easier to see the transitions of all the ACCESS bits using this format.
- 6. Examine the results waveforms by scrolling the Trace or List windows as appropriate. Note that the ACCESS bus is not counting up as expected.

Now let's load the add\_det design and create some tests to isolate the problem. Although there are many ways to approach circuit and waveform debugging, we will use just one for the purpose of this exercise.

7. Load the **c** add\_det design by choosing the **File** > Load > Design pulldown menu item and the Navigator.
- 8. Analyze the design and determine how to approach the problem by performing the following steps:
  - a. Open the schematic view window.
  - b. Maximize the schematic view window so that it fills the session by clicking on the maximize button. You will return this window to its original size in a later step.

  - d. Examine the design and notice that two 74259 addressable latches are driven by one 74LS161A counter. Therefore, we'll create a set of stimulus and assertions for the counter outputs: QA, QB, QC, and QD.

The stimulus you create can be used to both drive the two 74259 latches and to create the assertions. The assertions can be used to test the actual output of 74LS161A.

- 9. Create a bus for the QA, QB, QC, and QD outputs of 74LS161A by performing the following steps:
  - a. Select the nets connected to QA, QB, QC, and QD.
  - b. Click on the following palette icon:

The Define a New Bus dialog box appears.

The "Source Nets" are listed in the order in which you selected them. If your list does not match the one shown in the next step you may want to cancel the dialog box, unselect all the nets, and reselect in the following order: QA, QB, QC, then QD.



c. Complete the Define a New Bus dialog box as shown here:

The following message is displayed in the message area.

#### **İ** A new 4-bit bus named "dout" has been defined.

- 10. Return the schematic view to its original size by clicking on the maximize button.
- 11. Add an ascending count pattern to the new bus by performing the following steps:

a. Report the new bus by choosing the **Report** > **Setup** > **Bus Definitions** pulldown menu item.

The Buses report window appears.



- b. Unselect all objects.
- c. Select the "dout(3:0)" bus name in the Buses report window by clicking on the text.

Note: Because this bus does not exist on the schematic, you can't just select it in the schematic view and then operate on it.

d. Generate an ascending count pattern for the dout(3:0) bus by clicking on the **[Stimulus] Pattern Generator** palette icon.

The Pattern Generator dialog box appears.

e. Complete the Pattern Generator dialog box as shown:

T					
Pattern Generator					
Specify the type of pattern you want generated: 1. Click On					
00000         0100         0001         1110         1010           0001         0011         0010         1101         0101           0010         0010         0100         1011         1010           0010         0010         0100         1011         1010           00100         0001         0100         1011         1010           00101         00001         0000         0111         0101					
Incr/Deer Value Walking 1 Walking 0 Alternation	filled in				
Incr/Decr value walking 1 walking 0 Alternating					
Signal to generate pattern for dout(3:0)					
Initial value 0 Incr Decr ea	ch pattern by 1				
Radix of input values: 🔶 🔷	$\diamond  \diamond  \diamond$				
3. Fill in these values Hex Octal Binary Decimal Float Signed					
Start at time 250.1 Duration of each pattern 100	Force type				
Total number of patterns 55					
	Vired				
Duration of high-impedance regions	Charge				
before and after each pattern:					
Before 0 After 0	Clear old forces				
4. Click On OK Reset Cancel					

The following message is displayed in the message area:

#### **i** Generating an incrementing pattern for 'dout(3:0)' ... Done.

Four new waveforms are created in the forces waveform database. These represent the expected output of the 74LS161A as well as the input for the two 74259's. What are the new waveform's names? (Hint: look in the forces waveform report window.)



- 12. Ensure that the dout(3:0) bus name is still selected and trace the pattern created for the bus by using the Trace stroke.
- 13. View all the trace information by maximizing the Trace window and by using the Zoom out command to see the entire simulation.

The bus trace shows the repeating count pattern.

- 14. Create assertions for the QA, QB, QC, and QD outputs of the 74LS161A by performing the following steps:
  - a. Convert the pattern you just created and traced into assertions by clicking on the following palette icon:



The Convert Waveforms to Assertions dialog box appears.

b. Select the forces waveform database in the upper left corner of the Convert Waveforms to Assertions dialog box as shown here:

Convert V		
Waveform DBs		
aux forces results stimulus	>>>> S	

c. Fill in the Output File entry box at the bottom of the Convert Waveforms to Assertions dialog box as shown here:



The remainder of the settings should be left in their defaults. This is a straight forward conversion of the force events to expected results (assertions).

d. **OK** the dialog box.

During a later QuickSim II simulation run the assertions will be compared to the actual results produced by 74LS161A. This will allow you to isolate whether the 74LS161A or its inputs are the problem.

15. Exit your SimView session ("Without saving").

#### **Procedure 2: Analysis Using Assertion Tests**

1. Invoke QuickSim II on the add\_det design by issuing the following command:

```
your_path/training/qsim_n
shell> cd $HOME/training/qsim_n
shell> $MGC_HOME/bin/quicksim add_det
```

The QuickSim II session window appears.

- 2. Maximize the QuickSim II session window.
- 3. Open the schematic view window.
- 4. Load the stimulus that was used in the Module 8 batch simulation by entering the following:

Schematic view dofile add\_det/stimulus\_

5. Load the assertions you created in the previous SimView session by entering the following:

Schematic view dofile add\_det/asserts\_

6. Run the simulation for 5500 ns

The Info Messages report window lists how many assertion failures have occurred:



7. Report the Assertion Tests by clicking on the **ANALYZE** icon.

palette

TESTED?

The Assertion Tests report window is displayed.

8. View the assertion failures by performing the following steps:

<u>س</u>ر (ژ

a. Click on the ANALYZE

palette icon.

The View Assertion Failures dialog box appears.

Complete the dialog box as shown here:



All three waveform types of the four outputs for 74LS161A are traced:

• Results of compare: The assertion test names trace as /n\$x\_assert

- Expected: The assertion waveform names shown as asserts@@/n\$x
- Actual: The results waveform names take the form results@@/n\$x

The abundance of errors indicates that either one of the input waveforms is incorrect or that there is something wrong with the 74LS161A model.

Because the inputs are so simple, let's first look at the input waveforms.

- 9. Open a second Trace window by choosing the **Setup > Open New Window > Trace** pulldown menu item.
- 10. Select in the schematic view window the two nets that are used as inputs for the 74LS161A counter: START and PULSE.
- 11. Add the forces (Force Targets) waveforms for the selected nets by clicking



12. Examine the event edges of the forces in the Trace#2 window by first activating the Trace#2 window and then using the following features:



Notice the START waveform's falling edge at 200 ns and the rising edge at 250 ns. Again, under normal circumstances, you would need to research the 74LS161A device and your design criterion to determine that the pulse on the START waveform is 25 ns premature. You must now modify the START waveform.

- 13. To shift the START pulse 25 ns perform the following steps:
  - a. Reset the State using the **RESET...** palette button before attempting to modify a waveform with QuickSim II; you cannot modify waveforms at times prior to the current simulation time.
  - b. If you haven't already done so, zoom in on the area to be modified using the zoom stroke. This will help you to perform the edit.
  - c. Set the Waveform Editor's "Time grid spacing" to **5 ns** by clicking on



setup palette icon.

d. Shift the forces@@/START waveform pulse so that the falling edge occurs at 225 ns and rising edge occurs at 275 ns by using the



 $\overrightarrow{}$  palette icon.

(The method is similar to the one used in the lab exercise for Module 7, "Creating and Modifying Stimulus".

- 14. Verify that the change solved the problem by running the simulation for 5500 ns again.
- 15. The assertion tests are displayed again in the Trace window (you may need to pop the Trace#2 window to the background by using the ↓ stroke).

Notice that the errors no longer exist and that the ACCESS(15:0) bus appears to be counting and latching. You will perform further debugging of the add\_det design in later labs.

- 16. Continue to use various Waveform Editor and Debug Gates icons to examine edge deltas and edge locations, and to create and move trace cursors.
- 17. Exit QuickSim II.



LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 9 Solutions" on page A-17.

Then continue on to the next module.

#### **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 9 Solutions" on page A-17.

- 1. What does a red dot in a traced waveform indicate?
- 2. Assertion tests are a result of comparing waveforms from what two waveform databases?
  - •
  - •
- 3. What feature might you use to identify the exact time that a traced edge occurs?
  - •
  - •

#### **Module Summary**

In Module 9 you learned more about the Waveform Editor and Debug Gates palettes and the features that they provide to aid in examining results waveform data. You also learned about the Analyze palette and the waveform testing features. The waveform testing consists of assertions, results, and assertion tests.

Assertions are the expected results and can be created using the Assert command or you can convert waveforms. The assertions you create are compared against the actual results created during a simulation run.

Assertion tests are the comparisons and are represented in Trace or List windows as pass or failure. The assertion tests are displayed graphically in the Trace window for easy visibility.

More information about each of the palette icons is provided by choosing the **Help > On Palettes > Palette Descriptions** menu items.

# Module 10 Simulating Hierarchical Designs

Overview	10-2
Lesson	10-3
QuickSim II Features	10-4
Debug Hierarchy Palette	10-6
Examining Design Hierarchy	10-8
Latching Design Version	10-10
Changing Properties	10-12
Changing Selected Property Values	10-14
Selecting by Property	10-16
Changing Named Property Values	10-18
Changing a Model	10-20
Design Changes in QuickSim II	10-22
General Effects of Design Changes	10-24
Non-Connectivity Changes	10-26
Connectivity Changes	10-28
Back Annotation	10-30
Back Annotation Object Priority	10-32
Lab Preview	10-34
Lab Exercise	10-36
Procedure 1: Making Design Changes	10-36
Test Your Knowledge	10-45
Module Summary	10-46

#### **Overview**



#### Lesson

Upon completion of this module, you will be able to:

- Use the Design Hierarchy window to examine the design hierarchy structure and components
- Latch the current versions used in your design so that the design can be edited without affecting your frozen version.
- Change timing properties on a design instance using the Change Property at cursor method.
- Examine the annotations that you have made in your design using the Back Annotation window.
- Import back annotations from lower levels of hierarchy to a higher level design viewpoint so that you can use the changes in the higher level simulation.



You should allow approximately 1 hour and 20 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

## **QuickSim II Features**

QuickSim II provides:

- Logic analyzer-type trace viewing
- 12-state, interactive logic simulator
- Waveform Management (create, edit, analyze, save)
- Interactive control. Trace, list, monitor, stimulate for time, and set breakpoints
- Common simulation user interface -- SimView/UI Compliant with Motif standards
- Setup, stimulus, results, and states
- Incremental design capabilities
- Speed versus accuracy simulation trade-offs
- Mentor Graphics simulation model support
- Timing. Technology files can model loading and environmental effects (such as temperature, voltage, and process)

#### **QuickSim II Features**

QuickSim II is a 12-state, interactive logic simulator that allows you to verify the functionality of electronic designs produced with Design Architect, the System-1076 language, or a structured EDIF format netlist. QuickSim II gives you:

- Logic analyzer-type tracing that lets you examine the logic states of signals.
- Interactive control of the simulation that lets you trace, list, monitor, stimulate, spawn actions, or set breakpoints on any signal.
- Interactive waveform editing capabilities for creating, modifying, analyzing, or saving stimulus or results.
- SimView/UI user interface that is shared with other Mentor Graphics simulators and is compliant with Motif standards to provide a common look and feel.
- The ability to save setup conditions, stimulus, results, and particular states of the simulation so they can be restored at any time in the future.
- Incremental design capabilities that let you quickly update modified design components and change property values during the simulation, even if the change affects the design's connectivity or timing.
- The ability to perform simulations at varying combinations of speed versus accuracy to obtain peak simulation efficiency.
- The use of all Mentor Graphics simulation modeling methods, which include QuickPart Table models, QuickPart Schematic models, Memory Table models, System-1076 VHDL models, the LM-Family of hardware modelers, and Behavioral Language Models (BLMs).
- The use of Technology Files to model timing as a function of pin loading and environmental effects (such as temperature, voltage, and process), as well as other modeling capabilities.

# **Debug Hierarchy Palette**



#### **Debug Hierarchy Palette**

The Debug Hierarchy palette contains palette buttons that allow you to examine and to move around within the design hierarchy. The following is a list of operations that you can perform from this palette:

- **Hierarchy Window**. Lists the viewpoint name and component name. Displays the Design Hierarchy window that displays (in text form) the hierarchical structure of your design. This show parent/child relationships within the design structure. The Design Hierarchy window is shown on page 10-8
- Select/Report Connected. Selects the net or pins that are connected to the object selected. It is not always obvious as net go through bus rippers and traverse hierarchy, what is connected. This helps.
- Select/Report Contained. Reports all the hierarchical pins of an instance.
- View Connected Nets. Allows you to trace or list the nets connected to the selected pin or pins.
- Add Synonym. Allows you to provide a synonym name to an existing object. Since the system usually only recognizes objects, especially instances and nets, by their internal handles (I\$245, N\$1776), this allows you to give these objects recognizable names. Thereafter, the name can be used in place of the handle.

You can also use the Add Synonym button and accompanying dialog box to add the newly named objects to the Trace/List windows.

## **Examining Design Hierarchy**

Presents a text/graphical view of your design.

#### (Menu bar) > MGC > Design Management > Open Hierarchy Window



### **Examining Design Hierarchy**

The Design Hierarchy window contains a status area and two other windows. The status area gives you the current simulation viewpoint pathname, and also the design component that is currently being displayed.

The left window displays the hierarchy of your design. You can double-click on any object that contains a schematic and the instances below that level will be presented.

The menu contained in this window allows you to perform hierarchical operations.

#### **Latching Design Version**



Latching/Unlatching operations:

- Latch a viewpoint -- all objects are frozen
- Update latch -- unlatch and re-latch to newest
- Unlatch the design viewpoint
- Controlled latching -- latch specific objects
- View references/versions

### **Latching Design Version**

When creating a design, you may need to freeze or latch the design in its current state. This allows you to release the design for simulation or layout while still editing the design source. In this mode, updates can be made to the design base concurrently during the analysis or layout phase.

To accomplish this, you need to *latch* and then save the design viewpoint. Latching prevents the versions that are referenced by the viewpoint from being deleted or changed, and specifies that only these version be used with the viewpoint. At the same time, new versions of your design objects can evolve (using your update process) without affecting your latched objects. Each time you invoke an application on your design viewpoint, it uses the latched versions of the referenced objects.

You can latch a design viewpoint using DVE or QuickSim II. The latching capabilities let you:

- Latch a viewpoint. This freezes the version of every object referenced by the design viewpoint. These objects include components, models, interfaces, VHDL source, and back annotation objects.
- Update latch. This operation unlatches the design and re-latches using the newest versions.
- Unlatch the design viewpoint. This lets the design viewpoint use the newest version of each object every time the design viewpoint is opened.
- Controlled latching. This lets you specify components to latch or unlatch.
- View references/versions. This lets you examine the references and versions of some or all of the objects used by the design viewpoint.



For more information on latching and unlatching design objects referenced in the design viewpoint, refer to the *Design Viewpoint Editor User's and Reference Manual*.



## **Changing Properties**

- Use to perform "what if" simulations
- Examples of typical properties
  - Rise and Fall properties -- for adjusting delays
  - Physical properties -- Cap\_net or Temperature
  - Design parameters -- bus width, ASIC library
- Property changes are highlighted
  - Red on color monitors
  - Can be shown or hidden
- Property changes written to the highest-priority back annotation object -or- to specific BA object

#### **Changing Properties**

Changing properties allows you to rapidly perform "what if" simulations. Examples of typical properties you can change include the Rise and Fall properties for adjusting delays; physical properties, such as the Cap\_net or Temperature properties (which may affect technology file constraints and timing); and properties that affect design parameters, such as a parameter that defines bus width.

Property changes are highlighted on the schematic (in red on color monitors), and they can be shown or hidden. The hide operation must be performed in the Design Viewpoint Editor.

Note that the simulator normally writes all property changes to the highestpriority back annotation object. You can also write to another back annotation object by first selecting it in the Design Viewpoint window, or by providing an explicit path for to the "BA Name" entry in the Change Property dialog box.

Back annotation objects are design data objects that hold design property changes and are associated with the design viewpoint. The next subsection describes how the simulator interacts with back annotation objects.

## **Changing Selected Property Values**

Use this method when owner is known, but: property Name is not known (ref vs. inst) property is not visible (model property)



#### Also choose: Add > Property

#### **Delete > Property**

#### **Changing Selected Property Values**

Another method you can use to change a property is by selecting the property owner first, and then changing one of the listed properties. Once the property owner is identified, QuickSim II can present you with a complete list of accessible properties, as shown in the Change Properties dialog box list.

This method of changing properties is useful when either you don't know what the property name is, or the property is hidden (such as the model property), and you want to change its value.

To change a property using this method:

- 1. Select a design object that owns the property you wish to change.
- 2. Click on: **DESIGN CHG**  $\xrightarrow{10}$
- 3. Select a property from the list that you want to edit.
- 4. OK the dialog box. A new Change Property dialog box appears to allow changes to that specific property.
- 5. Enter the new value, the type, and the back annotation path.
- 6. OK the Change Property dialog box. The change is made and recorded in the specified back annotation object. The change appears in red.

## **Selecting by Property**

Use this method to work with multiple objects.

(Menu bar) Edit > Select > By Property...

Example: Select all inverters on the sheet:



All instances with model property "inv" are selected.

### **Selecting by Property**

There are times when it is advantageous to select a group of objects on which to perform a similar operation. Instead of performing the operation many times, you can select all of the objects at one time and perform the common operation once.

A useful way to select common objects is selection by property. Because properties are used by QuickSim II and other applications throughout EDA designs, you can also use them to help with manual editing functions.

For example, you have used generic components throughout your design and now want to replace the "0" timing values with real values. You can use the Select By Property operation to select all of the components that will get the same value for a property change. You then issue the Change Property operation, and all selected instances will reflect that change.

You can also use Select By Property to globally change the functional model type for a particular instance type. The example on the previous page shows that selecting the "model" property with a value of "inv" selects all generic inverters at the current level in the design. You can then use the Change Model operation to change all inverters at one time.

## **Changing Named Property Values**

- Use when you want a global replacement
- This dialog box appears when nothing is selected or many objects are selected

Click on:	DESIGN CHO	□ <sup>10</sup> ← 11 CHANGE PROPERTY			
Change Property					
Unable to display values because multiple objects are selected Highlighted property name will					
Existing Property Name: be used unle		be used unless	other property		
		/ name is speci	fied below.		
		Other Property Name			
New Property Value					
	Change on s	selected objects? Yes	No		
Property Type <ul> <li>string</li> <li>number</li> </ul>	Specify an Object Path: Specify an Object Path: Nets		Object Type Instances Nets		
			Pins		
✓ triplet					
	BA Name				
	ОК	Reset Cancel			

#### **Changing Named Property Values**

Because V8 designs use viewpoint environments, properties can be changed directly in QuickSim II and added to the back annotation file in the viewpoint. In addition, multiple back annotation files can be connected to a design, allowing flexibility in experimenting with the design.

You can make changes to multiple properties using the following method. For example, you used a generic library AND component in your design containing "0" Rise and Fall property delays. You now want to change these to real delay values. First you would select all of the AND gates.

To change design property, click on: **DESIGN CHG** 



The "long form" of the Change Property dialog box appears. If you did not select an object on which to modify properties, or you selected multiple objects, you will see the dialog box appear as shown.

Enter the property information in the dialog box, or choose from a list of properties in the lower dialog box. In the figure, we selected the "fall = 10" property, so the Change property: fall dialog box appeared. The existing value will show in the Value field. Change this value and **OK** the dialog box.

You can also and delete properties using **Add > Property** and **Delete > Property** from the popup menu.



For information about changing a design within an application, refer to the *Design Viewpoint Editor User's and Reference Manual*.

#### Changing a Model

Why change models?

- Change technology -- 1.0 micron to 0.8 micron
- Change model type -- QuickPart Table in place of sheet-based model (performance issue)

Remember: If it wasn't registered with the component interface, it can't be brought into the simulation

Model types and versions are set when the viewpoint is created (evaluation phase).

ARCH

To change a model: **DESIGN CHG** 

Change Model on Instance: /I\$1		
Models [Model Name, Model Type, [label1,, labeln]]		
["schematic", "mgc_schematic", ["\$schematic", "schem ["qpfile", "Tdm_qpt_do", ["\$gen_qpt"]]		
BA Name		
OK Reset Cancel		

#### Changing a Model

Changing a model is just a special case of changing a property. You could use the normal Change Property operation on the model property to change the model. A special Change Model command has been provided to allow you to see the models that are available before you make the change.

The previous page shows the Change Model on instance dialog box. It contains a list window that indicates which models are available. This instance has two models available; a schematic model with a label \$schematic and a QuickPart table model with a label of \$gen\_qpt. To specify a specific model instance model property is given the value of one of the available labels. Notice that the schematic model is shown as selected.



The selected model is not necessarily the current model. You must perform a report on the selected instance to determine the current model.

To change the value of the model property:

- 1. You bring up the Change Model dialog box as shown and select the model from the list.
- 2. Provide a pathname in the BA Name field if a back annotation object is not specified.
- 3. Then OK the dialog box.
- 4. Save the design viewpoint which saves your changes to the Model property. You do not have to re-invoke DVE to continue working with the design.



For information about model registration, refer to the "Registration and Labeling" section in the *Design Architect User's Manual*. Digital modeling methods and process are in the *Digital Modeling Guide*.

### **Design Changes in QuickSim II**

QuickSim II supports three types of design changes without leaving the simulator:

- Reloading models Access most recent model version
- Swapping models Changing the Model property
- Changing design properties (back annotating) -Add or change property values

After a change, simulator automatically validates related design information:

- Timing cache
- Saved state data objects
- Any displayed information
# **Design Changes in QuickSim II**

QuickSim II supports incremental design changes, which can decrease the time for a design iteration (the cycle of design creation, revision, and simulation), depending on how much of the design is actually affected. You can make three basic types of design changes without leaving the simulator environment:

- **Reloading models.** You can reload models to obtain the most recent model version. For example, assume you invoke the simulator on a large design and, after simulating for a while, you discover a problem with the schematic or a technology file. You can correct the problem using the appropriate editor, recompile if necessary, and reload the result into the simulation without having to exit and re-invoke the simulator.
- **Swapping models.** You can swap models by substituting one model representation for another. This is done by changing the Model property. For example, if you want to use a gate representation in place of a VHDL representation, you can change the Model property from within the simulator to refer to the schematic.
- **Changing design properties.** You can add or change property values directly in the simulator, referred to as *annotating the design*.

The simulator keeps track of all incremental design changes. This ensures compatibility with related design information that is saved, such as the timing cache and save state data objects. The simulator automatically checks the data objects that are dependent on the design configuration, and prohibits them from being used if they are not compatible.



For a complete description, refer to the "Digital Model Organization and Evaluation" section of the *Digital Modeling Guide*.

### **General Effects of Design Changes**

- Timing values are recalculated, if necessary, and appended to the timing cache
- Displayed timing is invalidated (lined out)
- Stimulus and setup is maintained so you can immediately run another simulation
- If a property is changed, the Back Annotation window is automatically updated
- Changes are highlighted (red on color monitors)
- Design status windows are not validated. You must recreate these windows to see valid data
- Restoring an existing simulation state is not allowed
- Saving simulation state is not allowed until you save design changes

# **General Effects of Design Changes**

The simulator responds to changes, whether or not a change affects the design connectivity. The following applies regardless of the type of design change:

- The potentially affected timing values are recalculated, if necessary. Instead of completely replacing the timing cache, the newly calculated timing is appended to the timing cache that was previously saved.
- All displayed timing information is invalidated. Windows that contain this invalid information, such as Timing Info windows, are lined out (diagonal lines). Use the update button to update their contents.
- The stimulus and the setup conditions are maintained. This allows you to immediately run another simulation. For example, your Trace and List windows, forces, and hierarchical checking and mode settings are maintained.
- If a property is changed, the Back Annotation window is automatically updated. This window shows you the changes that currently exist in the back annotation object.
- **Changes are highlighted.** If a visible property is changed on a currently viewed design item, the new value is highlighted (in red on color monitors).
- **Design status windows, (Object or Parts List windows) are not affected.** If you want new status information, update or recreate these windows.
- **Restoring an existing simulation state is not allowed.** Any simulation state that exists at the time you change the design becomes invalid.
- Saving the simulation state is not allowed until you save the current design state. The simulation data that is written must be associated with a specific, persistent (saved on disk) version of the design viewpoint and the back annotation object.

# **Non-Connectivity Changes**

### Examples of non-connectivity changes

- Rise and Fall properties
- Modelfile properties
- Net delay, decay and init properties

What happens?

- The simulation time is not affected
- Windows displaying signal activity are not affected
- New events will use the new timing values and modelfile data
- Since pending events use old timing values, a reset state is recommended

## **Non-Connectivity Changes**

The simulator handles design changes differently, depending on whether the change affects the *design connectivity*. Design connectivity refers to the way the nets, pins, and instances are connected or related, and is always determined in the context of the design viewpoint.

**Non-connectivity changes.** Most property changes do not affect design connectivity, such as changes to Rise and Fall properties. The exceptions are changes to Model properties and properties that are used in frame or Model property expressions.

When a change is made that does not change design connectivity, the following additional behavior applies:

- The simulation time is not affected. Although you can continue your simulation, your results may not be easy to understand, depending on the nature of the change and the design being simulated. For example, any events that are pending when the change is made are scheduled using the old delay values. Therefore, it is recommended that you use the Reset State command (or the **Run > Reset** pulldown menu item) to reset simulation time to zero and initialize the design before continuing.
- Windows that display signal activity (such as the Trace, List, and Monitor windows) are not affected. The data displayed in windows remains intact.

# **Connectivity Changes**

Connectivity has changed when:

- Model is reloaded (newer version)
- Model property is changed
- Frame expression is changed

What happens when connectivity changes?

- Source view windows are updated (lines out or deletes non-existent views)
- Simulation time is automatically reset to zero
  - Clears all report window displays
  - Re-initializes the design
- Signal names in keep list and forces waveform database are verified
  - Signals validated
  - Stimulus of invalid signal disconnected
  - Invalid signals removed from keep list
  - Invalid expressions and breakpoints deleted
  - Status window lists the removed signals

## **Connectivity Changes**

The simulator always considers the connectivity of the design to be affected when you either reload a new version of a model or you change the value of a Model property (swap a model). Also, changes to properties that are used in frame expressions or Model property expressions affect design connectivity.

To maintain a reliable simulation, the simulator uses a pessimistic model when deciding the extent of change to the connectivity of the design.

When a change is made to the design connectivity, the following additional behavior applies:

- Source view windows are updated. The schematic view and VHDL View windows automatically reflect the new model versions. This behavior might delete an entire schematic view or VHDL View window, if it contains data that no longer exists in the new version. (As an alternative to deleting VHDL View windows, you can instruct the simulator to merely line it out.)
- The simulation time is automatically reset to zero. This clears all report window displays and re-initializes the design.
- Signal names in the keep list and the forces waveform database are verified. Signal names are invalid when the signal they refer to is no longer in the design. Stimulus applied to invalid signals is disconnected (although their waveforms remain in their respective waveform databases). Also, invalid signals are removed from the keep list and any windows they may appear in, and expressions and breakpoints that reference the invalid signal names are deleted. The simulator lists in a status window any signals that are removed.



For more information on the effects of connectivity changes on QuickSim II, refer to the *QuickSim II User's Manual*.

# **Back Annotation**

### • Back Annotation connections--via the viewpoint



• Back Annotation Window

		В	Back Annotation: rev_2	2		
Í	In	stance Pathname	e (Property Name, Prope	rty Value)		
		/reg/l\$3/l\$10 /reg/l\$5/l\$11/l\$25	GROUP_SEED BRD_LOC	seed_value 245, 549, 1,	90	
	Pi	in Pathname	(Property Name, Property	Value		
		/reg/I\$3/P\$23	DRIVE	SSZ		
	N	et Pathname	(Property Name, Property	Value)		
		/reg/l\$4/N\$27	NET_LENGTH	241		

# **Back Annotation**

Back annotation is the term used for adding, changing, and deleting properties by placing changes in a back annotation object rather than directly changing the design. Back annotations are the only way you can make design changes directly in QuickSim II. Otherwise, you must use Design Architect to make direct design changes.

When you are working with multiple back annotation objects, you need to be aware of which annotation object is storing the edits. To help you maintain and organize your back annotations into specific groups of changes, DVE allows you to: connect, disconnect, import, export, prioritize, and edit back annotation objects. The following discuss the processes used in managing back annotation objects.

When applying property edits to a design through back annotation, the application needs to know which back annotation object should receive the edit. This back annotation object is called the "active" back annotation object. The "active" back annotation object receives all property edits until a new (different) back annotation object is specified. There are four ways you can specify a back annotation object as "active" in DVE:

- 1. Supply a different back annotation name when executing a back annotation menu item, command, or function.
- 2. Interactively select a specific back annotation object in the Design Viewpoint window.
- 3. Make a specific Back Annotation window active and issue a command.
- 4. If no back annotation object has been specified during this session, the back annotation object with the priority of "1" (highest) is active.

# **Back Annotation Object Priority**

- Last connection made has highest priority (1)
- Change priority by connecting/disconnecting back annotation objects (using DVE) or specify (connect) a new object in QuickSim II

Example: Current priority is A, B, C. Change to C, B.

Priority	Original	Disconnect A&C	Connect C
1	Back Annotation Object A	Back Annotation Object B	Back Annotation Object C
2	Back Annotation Object B		Back Annotation Object B
3	Back Annotation Object C		

## **Back Annotation Object Priority**

When multiple back annotations are attached to a design viewpoint, they are assigned priorities, with 1 being the highest priority. If the same property is changed in more than one back annotation object, the change specified in the highest priority back annotation object supersedes all others.

To change the priority of back annotation objects, you need to disconnect and then connect them in a particular order to set your new priority. To disconnect a back annotation, choose the **Design Viewpoint > Disconnect Back Annotation** menu item. To connect a back annotation, choose the **Design Viewpoint > Connect Back Annotation** menu item.

For example, assume three back annotation objects (A, B, and C) are connected to the design viewpoint, with A being the highest priority. To make the priority C, B, with A disconnected, do the following as shown in the previous table:

- 1. Disconnect objects A and C. B is now the only connected back annotation object (and has a priority of 1).
- 2. Connect object C. The priority is C, B, with C now having priority 1.

When you view the list of connected back annotation objects in the Design Viewpoint window, the object with the highest priority is numbered "1", then "2", and so on to the lowest priority object.

In order to use back annotations, you must have at least one back annotation object connected to the design viewpoint. If you currently have a back annotation object connected to your design and you connect an additional back annotation object, both remain connected with the *new* back annotation object having the highest priority. All back annotation objects remain connected until they are disconnected.

# Lab Preview

- Invoke QuickSim II on add\_det
- Use selection by property to select groups of objects
- Change Rise and Fall property values
- Change a model property value
- Add the Ref (reference) property to an instance
- Add a property with an expression value
- Specify and examine back annotation objects
- Save the back annotation object with the viewpoint

### Lab Preview

In the lab exercise for this module, you will:

- Change Rise and Fall property values for generic library components using the global select and replace method.
- Change Qbrise and Qbfall property values on a single instance using the change property at cursor method.
- Change the model property of an instance during a simulation run, and determine the changes that occurred to the simulation environment.
- Add the Ref (reference) property to an instance.
- Add a property with an expression value.
- Examine a back annotation object using the report window.
- Save a back annotation object to the design viewpoint.

### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

### **Procedure 1: Making Design Changes**

- 1. If necessary, log into your workstation and set your current directory to your qsim\_n directory.
- 2. Invoke QuickSim II on the design using default shell invocation, as follows:



- 3. Open a sheet for the add\_det design and maximize the window to fill the session area.
- 4. Load and connect the forces waveform database:

Issue the command: dof add\_det/lab6\_forcefile

5. Run the simulation for 3000 nsec.

This run will be used to illustrate what happens when you make design changes within a simulation run.

- 6. Select the type-D flip flop instance and change the instance model by doing the following steps:
  - a. First, select the type-D flip flop instance (at the top of the schematic view window.
  - b. Click on: DESIGN CHG

The Change Model dialog box is displayed. Although the first line in the model list area is selected, this may NOT be the current model. To discover the current model:

- c. First **Cancel** the Change Model dialog box.
- d. With the dff instance still selected, use the r stroke to generate a report on the selected object. ►
- e. Examine the "Properties" list. The current "model" property is set to "\$gen\_qpt" which indicates that a QuickPart Table model is being used.
- f. Now close this Objects report window.
- g. With the dff instance still selected, again click on: **DESIGN CHG**
- h. Make sure that the following line is selected: ["schematic","mgc\_schematic"'["\$schematic","schematic","default"]]
- i. **OK** the dialog box.

NOTE: The BA Message window may appear with the message: "your\_path/*training/qsim\_n/LATCH/default* is not attached to the current viewpoint. Would you like to create/attach it?" If it does, click on the **YES** button.

The Message window displays "Stopped at 0.0ns" which means that the simulator was reset to time zero. This happens whenever connectivity (models) is changed.

7. Verify that the dff model has been changed as follows:

You will again examine the design object report to determine if the model change has actually been made. Make sure that the dff instance is selected.

a. Use the r stroke again to generate a report on the selected object

Note that the model is now listed as "schematic" instead of \$gen\_qpt. Also notice the "Primitive Instance" is set to "FALSE". This is because schematic models are hierarchical and thus not primitives.

- b. Close the Objects report window.
- 8. Change the Rise and Fall delays for the D-flipflop component as follows:
  - a. Use the  $\checkmark$  stroke to zoom in on the QB output pin of the D-flipflop so that you can easily see the properties below the output.
  - b. Click on: **DESIGN CHG**



- c. Place the crosshairs on the top (Qbrise) property and change its value to: "6 13 25" (triplet value--don't use commas, only spaces).
- d. Repeat the Change Property @ Cursor operation to change the bottom (Qbfall) property to: "13 25 40"

Note: You do not need to change the properties for the Q output, since it is not connected.

- 9. Change the Rise and Fall properties on all four inverters as follows:
  - a. Unselect everything (use the  $| \downarrow |$  stroke).
  - b. Choose: (popup) > Select > by Property
  - c. When the Select By Property dialog box appears, fill it out to resemble the following figure:

Select By Property				
Property Name	model	Property Type String		
Property Value	inv	<ul><li>✓ Number</li><li>✓ Regular Exp</li></ul>		
Select: Other Types				
Nets	Monitor_Fla Group	ag 🛆		
Pins	Bus Time			
ОК	Reset	Cancel		

d. **OK** the Select By Property dialog box.

This selects all the inverter instances, but the Rise and Fall properties, which are attached to the output pin, are not selected. You will select them next.

e. Click on: DBG GATES



#### > Primitive > Driving

Now only the output pin of each inverter is selected, and you are ready to change the properties.

Œ

- f. Click on: DESIGN CHG
- g. When the Change Property dialog box appears, fill in the following:

Select "Existing Property Name:" **rise** New Property Value **4 9 15** Property Type **triplet** 

h. OK the Change Property dialog box.

The Rise property on the 4 inverters has changed to the new value

i. Do NOT unselect the selected pins. Use steps f and g again to change the Fall property to the following:

New Property Value 5 10 15

j. Verify that all of the inverters have new Rise and Fall values.

10. Use the procedure in the previous step to change the Rise and Fall properties on the output of the three AND gates (model = AND) to the following values:

Rise = **4 8 15** Fall = **5 10 20** 

Examine the AND gate to verify that the properties have been assigned correctly. When you have completed all of the timing changes, your design should contain the following values:



11. Examine the contents of the back annotation object as follows:

#### a. Choose: **Report > Design Changes**

The Back Annotation window appears. There are three groups of rows with a heading above each row. What are the three field headings it contains?

1. \_\_\_\_\_ 2. \_\_\_\_ 3. \_\_\_\_

Note that the changes you made appears in the Back Annotation object. Which change appears beneath the "Instance Pathname" heading?

What type of changes appear beneath the "Pin Pathname" heading?

- b. Do not close the Back Annotation window yet, but instead make it an icon to be used later.
- 12. Add a reference designator (Ref property) to the instance as follows:
  - a. Select the D-flipflop.
  - b. Click on: DESIGN CHG



c. Enter the following in the dialog box and **OK** it:

(New Property Name)ref(Property Value)dff\_1A(Property Type)string

- 13. Add a property to the output of the D-flipflop as follows:
  - a. First, unselect all objects.
  - b. Select the portout symbol for the FULL signal.

The net connected to this symbol is selected. This net also connects to the QB output of the type-D flip flop.

c. Click on: DESIGN CHG



The Add Property dialog box appears which allows you to add the property/value combination.

d. Enter the following:

(New Property Name)	FANOUT
(Property Value)	10*TCAP
(Property Type)	expression

- e. Now **OK** the dialog box.
- f. Double-click on the Back Annotation icon to view the change.

Which entry heading does the new change appear beneath?

What is the value for this property?

g. Close the Back Annotation window.

- 14. Save the design changes to the viewpoint as follows:
  - a. Click on: DESIGN CHG



Because you haven't saved the information yet, a question box appears asking if you want to save the current design changes.



b. Click on: Yes

Your design changes (BA object) are saved to the design viewpoint. Also notice that an Info Messages window reports that your Timing Cache was updated with the new timing changes.

15. Exit QuickSim II.



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 10 Solutions" on page A-18.

Then continue on to the next module.

# **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 10 Solutions" on page A-18.

- 1. List two ways to change a model in QuickSim II.
  - 1.\_\_\_\_\_ 2.\_\_\_\_
- 2. What determines the back annotation object priority when more than one BAO is connected?
- 3. Name the benefits that the design viewpoint provides:
  - 1.\_\_\_\_\_ 2.\_\_\_\_

# **Module Summary**

The design viewpoint can contain back annotation objects. These objects hold the changes that are made to the design in QuickSim II. Back annotation objects are versioned, and can be connected to or disconnected from the viewpoint using DVE. A priority scheme determines which back annotation object is used when more than one is connected.

# Module 11 Debugging Design Analysis

Overview	11-2	
Lesson	11-3	
Debug Gates Palette	11-4	
Using Probes	11-6	
Locating Unknown (X) Sources	11-8	
Design Changes Palette (SimView)	11-10	
Design Timing Constraints	11-12	
Setting Instance Constraint Mode	11-14	
Spike Models	11-16	
X-Immediate Spike Model	11-18	
Suppress Spike Model	11-20	
Technology File Spike Model	11-22	
Changing Instance Contention Models	11-24	
Conditions That Affect Kernel Setup	11-26	
Lab Preview	11-28	
Lab Exercise	11-30	
Procedure 1: Setting Up the Debug Environment	11-30	
Procedure 2: Fixing Constraint Violations	11-36	
Test Your Knowledge	11-43	
Module Summary	11-44	

### **Overview**



### Lesson

Upon completion of this module, you will be able to:

- Use the Debug palette to perform debugging operations within the QuickSim II simulation run.
- Create a probe on an unnamed net, and assign a name (synonym) to that net.
- Use the Unselect Except X and Backtrace Nets=X palette buttons to find and locate the source of unknown signals in your design.
- Turn on constraint checking in your design, and understand what it means when you have a constraint violations.
- Change the spike model within the QuickSim II kernel, and understand how each model affects your results.
- Turn on spike checking with a digital simulation.
- Set the contention model within your design, and understand how each of the available models work.
- Turn on contention checking within a digital simulation.



You should allow approximately 1 hour and 25 minutes to complete the Lesson, Lab Exercise, and Test Your Knowledge portions of this module.

# **Debug Gates Palette**

- Create/Manipulate Trace window cursors
- Add a Probe or Monitor Window



### **Debug Gates Palette**

The Debug Gates palette is displayed when the DBG GATES button is chosen. This palette contains icons that assist you in analyzing a circuit after a simulation in which results were obtained.

Monitoring and probing operations are also contained on this palette. With them you can quickly display signals that need further study.

When working in a Trace window, you can use the cursor manipulation icons to add, move and snap cursors. The cursor is a useful method of viewing the state at any point along a trace.



For more information about the function of each icon button in this palette, choose the (Menu bar) > Help > On Palettes > Debug Gates menu item.

## **Using Probes**

### A probe adds a synonym name to a net



To delete a probe, choose menu item: Delete > Probe

# **Using Probes**

A probe is a graphical synonym. The probe not only creates a synonym for the object, but places a visual graphical reminder on the schematic sheet, called a flag and (when operating SimView/UI with a simulator) performs an implicit keep on that object.

To create a probe, you click on the **[Debug Gates] Add Probe** palette icon. A dialog box appears to help you define the probed object. This dialog box is shown on the previous page.

As the dialog box shows, you can add probe-type synonyms to instances, nets, or pins. To create the probe, you can either select the object to be probed, or supply the name of the object. The name, in many cases may be a net path such as /I\$23/I\$221/N\$414. You can see that selection might be preferred to typing these long paths.

If you click on the "Location" button, you can add the probe to an object *after* executing the dialog box. This method does not use the standard select-then-act method used by other operations. A set of crosshairs allows you to pinpoint the object that receives the probe. If you have chosen "Any" for the type, the nearest object receives the probe. If you filter the object type (such as "Net") the nearest object of that type is assigned.

Because a probe is just a graphical synonym, you can assign it to instances, nets, or pins. Each of these objects has a hierarchical path and *handle* within the design. You are actually creating a synonym (and a keep) for that object path.

To delete a probe, you are not allowed to simply select it and delete the selected object. You must use the **Delete > Probes** menu item. A dialog box appears that allows you to name the probe to detect, or to select "All" which will delete the synonyms and remove the graphical objects.

You can also use the Add Synonym command, which will create flagged or unflagged synonyms, but no keep is performed.

The **Report** > **Setup** > **Probes** menu item reports all Probes. The Report Synonym command reports all synonyms.

## Locating Unknown (X) Sources

Allows you to detect and examine unknown signals.

**FXCEPT X** 

- 1. Access DEBUG GATES palette: DBG GATES
- 2. Select all nets: Edit > Select > All > Nets
- 3. Select a net with "X" state X. -D X. UNSELECT





Tries to determine the input that caused the X Unselects the output net & selects input X net

1. "Backtrace succeeded."

You repeat this process until you cannot go further

**2.** 'Backtrace complete. There are no more nets in this path that have an 'X' state value.''

Examine the instance(s) where this occurs

# Locating Unknown (X) Sources

If your design is producing X signal states, you can use the Debug Gates palette to find the instance that is generating the X state. The BACKTRACE NETS = X icon allows your to search back through a circuit for the cause of an X value. To back trace X signal states, perform the following steps:

- 1. Activate the Debug Gates palette by clicking on the DEBUG GATES palette selection button.
- 2. Select the net whose value is X.

Note that back tracing X states works best when only one net is selected. With only one net selected, you can focus on a single path.

3. Click on the BACKTRACE NETS = X icon.

The simulator examines the inputs of the instance that is driving the selected net. If the signal value of any input to the instance is X, the simulator selects the net attached to the input, unselects the initially selected net, and displays the following message in the Messages area:

Backtrace succeeded.

You can then continue back tracing by clicking on the palette icon again. Note that if the simulator selects more than one input to the instance, you may want to unselect all nets except one so that you can focus on a single path. You can repeat the procedure to back trace the other paths.

If none of the inputs to the instance have a value of X, the simulator does not unselect any nets and displays the following message in the Messages area:

Backtrace complete. There are no more nets in this path that have an 'X' state value.

You can now examine the instance that is attached to the selected nets to see why it is generating the X state.

# **Design Changes Palette (SimView)**

• Work with back annotation information



## **Design Changes Palette**

The Design Changes palette is displayed when the DESIGN CHG button in the palette selection area is chosen. This palette contains icons that allow you to add, delete, or modify properties (including the model property). Since the back annotation object is the mechanism for storing property changes, this palette also allows you to manipulate back annotation objects.

Design changes can be made directly in SimView, without requiring you to exit and rebuild the database. In addition, component models can be changed in SimView, as long as the change uses the current version of the component interface.



For more information about the function of each icon button in this palette, choose the (Menu bar) > Help > On Palettes > Design Changes menu item.

# **Design Timing Constraints**

- Checks one signal against other input signals
- Specified in Technology Files
- Can be of the following type
  - Setup (tS) Signals must be stable before this signal changes state.
     Used for clocking in data.
  - Hold (tH)
    - Signal must "hold" state for period
    - Used to hold data after write operation
  - Width (tW) Specifies minimum pulse width.
     Used on latching pulses
  - Fmax Specifies instantaneous frequency
  - Stab Signal is stable compared to other signal
# **Design Timing Constraints**

Design timing constraints allow the simulator to check to determine if your design is operating within a predetermined set of timing relationships. Many of the components that you use in your design already have these constraints defined.

Constraints allow the simulator to check a change on one signal against other related signals. For example, data and address must be stable on memory before a write cycle can begin. A setup constraint would determine how long the data must be present before the write line is strobed. You must also keep the data and address there until the write is finished. A hold constraint determines how long after and event that the data does not change.

The following is a list of constraints that QuickSim II can check during a simulation run:

- Setup (tS) Signals must be stable before this signal changes state. As in the previous example, setup is used on data signals compared to the clocking signal for the data.
- Hold (tH) Signal must "hold" state for period. The hold constraint is used to verify that address information is stable until a data read or write operation has completed.
- Width (tW) Specifies minimum pulse width. Used on any device that requires a minimum pulse duration to operate properly. This specification might be placed on a RESET signal. A spike could also be generated if a signal was too short.
- **Fmin/Fmax** Specifies the minimum or maximum instantaneous frequency allowed. The simulator actually calculates the positive and negative pulses in the signal and reports a warning if the pulse width is too long/short.
- **Stab** Signal is stable compared to other signal. Signals should not change during the stable period.

#### **Setting Instance Constraint Mode**

Constraints - setup, hold, width, fmax, stab

```
Example:
setup = 1.0, 2.0, 3.0 on 'A'(V) to WRITE_EN (AL)
do { sim_$send_msg("Error: setup violation at
time %1S on instance %2S on pin %3S",
sim_$time(),sim_$inst_name(),
sim $on pin name()); };
```

SETUP	Change Co On Selected instan Instance name	Change Constraint Mode On Selected instances Named instances Instance name		
	Constraint mode Off Set state Messages	_ Override		
	OK Reset	Cancel Help		

- "Off" No constraint checking (default)
- "Set state" Sets output state on violation
   No messages are displayed
  - 0 No messages are displayed
- "Messages" sets state and displays messages

#### **Setting Instance Constraint Mode**

Design constraints are device limitations you define in technology files. During the simulation, the simulator can verify if these constraints are being met, and if they are not, it can respond with an appropriate warning message. There are three constraint checking modes:

- **Off.** Default when the simulator is invoked. No constraint checking.
- Set state. Enables timing constraint checking. Sets the state of the model's output pins when a violation occurs. No constraint violation messages are displayed.
- **Messages**. Checks for all constraint violations, sets the output pin state, and displays appropriate messages when violations occur.

You can set the constraint mode at any level in the design hierarchy, or for the entire design.

During a simulation session, you can enable constraint checking for specific instances at any design level. The following procedure describes how to enable constraint checking for one or more instances:

- 1. Select the desired instance or instances. Although selecting an instance is not required, it is generally easier than specifying names of instances.
- 2. Click on the [Setup palette] Constraint button to display a dialog box that prompts you for the constraint mode you want.
  - Enable state setting
  - Enable message display
  - Force the constraint mode to the design levels below the selected instances
  - Specify names of instances if you do not select instances
- 3. Activate your choices by clicking the **OK** button.

OUT

# Spike Models



What causes spikes?

IN

- Spike condition -- simulator schedules an event on a pin that already has an event scheduled
- Usually occurs when input pulse is shorter than the input to output delay.

How do you set the spike model (what happens when a spike occurs):

- Set the "kernel" spike model:
  - Spike suppress model
  - X-immediate model
- Specify a Technology File spike model (overrides kernel model)

# **Spike Models**

A simulation spike attempts to model the design condition where pulses of very short duration are traveling through your circuit. The behavior of these short pulses in real hardware can vary considerably. QuickSim II detects short pulses and gives you several ways of handling the pulse.

A pulse is considered short when the pulse width is shorter than the propagation delay of the gate handling the pulse. For example, the buffer on the previous page has a pulse train at the input with transitions 8ns apart. The gate output delay is 7ns for a rising edge, and 9 nanoseconds for a falling edge.

The first rising edge schedules the output event 7ns later (rise value = 7). Since the next input edge is not until 8ns later, this event is already processed and does not cause a spike condition.

The negative-going edge schedules an output event 9ns later (fall value = 9). Since the next input edge is 8ns later, this scheduled event has not yet matured (still pending) when the rising edge occurs. Therefore a spike condition is detected. The spike is handled in one of two ways:

- **X-Immediate model**. The output goes to an unknown state. This model is examined in more detail on page 11-18.
- **Suppress model**. The event that the spike would have caused is suppressed so an unknown state is not created. More information on this spike model type is on page 11-20.

You control what type of model is used for the instances in your design. As you saw on page page 5-12, the Setup Analysis dialog box allows you to specify this model globally for your design. You can also use the individual **Setup** > **Kernel** > **Change** > **Spike model** dialog box to change the spike model for individual instances in your design.

# X-Immediate Spike Model

- Current value (1,0,X) not = scheduled logic value & scheduled value not = violating state value Spike state is X, otherwise same as current
- Current strength (S,R,Z,I) not = scheduled strength & scheduled strength not = violating state strength Spike strength is I, otherwise same as current



What happens:

- 1. Simulator removes scheduled event (gray)
- 2. Simulator schedules spike state with no delay (next iteration)
- 3. Simulator schedules new state with delay

#### **X-Immediate Spike Models**

Spike models instruct the simulator on how to handle spike conditions. A spike condition is a violation that occurs when the simulator tries to schedule an event on a pin that already has an event scheduled. When a spike occurs, three signal values are at play: the current driving state of the pin, the scheduled state, and the violating (or "new") state. The simulator can use two types of spike models: the X-immediate model and the Suppress model (discussed on the next page).

- X-immediate model follows these rules:
  - a. If the logic value (1, 0, X) of the current pin state doesn't equal the logic value of the scheduled state, and the logic value of the scheduled state does not equal the logic value of the violating state, the logic value of the spike state is X. Otherwise, the logic value of the spike state is the same as the current state.
  - b. If the strength (S, R, Z, I) of the current pin state is not equal to the strength of the scheduled state, and the strength of the scheduled state is not equal to the strength of the violating state, the strength of the spike state is I (indeterminate). Otherwise, the strength of the spike state is the same as the current strength.
  - c. The simulator removes the scheduled state from the event queue.
  - d. The simulator schedules the spike state with no delay.
  - e. The simulator schedules the new state with any delay that is associated.

Note that the spike state starts one iteration after the simulator detects the spike and stays on the signal for the duration of the spike condition. (The spike duration equals the scheduled time of the violating state minus the current simulation time.)

This spike model is considered pessimistic because it assumes that the new state has a negative effect on the resulting output.

## **Suppress Spike Model**

- Suppress model description
  - Simulator removes scheduled state
  - Simulator adds new state to event queue
- This model is considered optimistic



Setting instance(s) spike model:



### **Suppress Spike Model**

The suppress spike model follows these rules:

- 1. The simulator removes the scheduled state from the event queue.
- 2. The simulator schedules the new state according to the associated delay. The simulation then continues as normal.

This spike model is considered optimistic because it assumes that the new state does not have a negative effect on the resulting output.

To receive notification of spike conditions, you report spikes using the Change Spike Check command or the **Setup > Kernel > Change > Spike Check** pulldown menu.

You can set the spike model on any instance or for the entire design. The following procedure describes how to set the spike model:

- 1. Choose the **Setup > Kernel > Change > Spike Model** pulldown menu item to display a dialog box that prompts you for the spike model you want.
- 2. Complete the dialog box. You can choose the X-immediate or the Suppress spike model. The "Override" button allows you to force the spike model to the design levels below the specified instances.
- 3. Activate your choices by clicking the OK button.



For more information about how these spike models affect the simulator, refer to the *Digital Simulators Reference Manual*. For more information about defining spike models using technology files, refer to the *Technology File Development Manual*.

# **Technology File Spike Model**

- Customizes spike model for each component
- This model overrides the kernel spike model
- Model defines 3 width-dependent spike regions:

suppr	ess region	X-pulse ı	region	transp	oort region
0	suppre	ss_limit	x_li	mit	total path delay
time (a	fter leading	pulse edge	e) ——		

#### **Technology File Example:**

```
SPIKE_MODEL MODEL_DEFAULT = {
   SUPPRESS_PERCENTAGE 40;
   X_PERCENTAGE 70;
   };

SPIKE_MODEL low_pulse(pth_del, i_pin) = {
   SUPPRESS_LIMIT (( pth_del * .133) + cp(i_pin);
        X_LIMIT (( pth_del * .276) + cp(i_pin);
   };

BEGIN
   tP 11, 13, 19 on IN(AL) to OUT(AL)
        SPIKE MODEL low pulse;
```

## **Technology File Spike Model**

The configurable spike model allows the modeler to specify three different regions in the period between the arrival of the previously scheduled output event, and the arrival of the conflicting event. The regions are "suppress", "X-pulse" and "transport". They are specified using two parameters, a "SUPPRESS\_LIMIT" and an "X\_LIMIT", which divide the spike period as shown in the figure on the previous page.

CONDITION	REGION	ACTION
pulse_width < suppress_limit	suppress region	suppress pending action, schedule new state if different from current state
suppress_limit <= pulse_width < x_limit	X-pulse region	an "X" state is propagated to the
		output until new state arrives
x_limit <= pulse_width < path delay	transport region	the pulse is passed through model
		to the output

A signal pulse on a model's output pin is handled as follows:

Spike models are defined in the DECLARE region of the model's Technology File. Once declared, a SPIKE\_MODEL may be explicitly associated with one or more propagation delay statements (tP) in the body of the timing model. There is no limit on how many spike models may be defined in a timing model, there may be one for every delay path if necessary.

The syntax for defining a spike model in a Technology File is given below.

```
SPIKE_MODEL MODEL_DEFAULT | NETDELAY_DEFAULT | <model name>
( <optional arguments> ) = {
[<optional directives>]
SUPPRESS_LIMIT <time spec> | SUPPRESS_PERCENTAGE <percentage>
X_LIMIT <time spec> | X_PERCENTAGE <percentage>;
};
```



For Spike\_model syntax, refer to "SPIKE\_MODEL" in the *Technology File Development Manual*.

#### **Changing Instance Contention Models**

- Contention: Two or more output pins drive a net
  - High impedance drivers (Z) are ignored

CON- TENTION	Change Contention Model
	On Selected objects Named objects
	Time 0 (Contention can exist this long before warning is issued)
	Model type
	None Any Same Different
	Driven 0 <- Contention pairs (See below for examples)
	Note: The model types above are ignored if a 'driven' value is given         The following are some examples of legal contention pair syntax:         Pair       Definition of when contention exists
	(1s & 0s) Driven by a '1s' and '0s' (1sr & 0s) Driven by a ('1s' or '1r') and '0s' (?s & ?r) driven by any 'strong' and any 'resistive' (0* & 1Z) Driven by any 'low' and '1z'
	OK Reset Cancel Help

- Model: None, Any, Same, Different
- Time is how long condition exists before "flagged"
- "Driven": Customize your own contention model

### **Changing Instance Contention Models**

You can direct the simulator to check for multi-driver signal contentions by attaching the Contention property to targeted nets, and then enabling contention checking. If the simulator finds that a contention condition exists, it generates an error message citing the pins that are responsible and the net where it occurred.

The Contention property defines the *contention model* that the simulator uses when it checks for contention conditions. You can assign one of four values to the Contention property, which in turn enables one of four contention models:

- None. Disables contention checking.
- Any. Contention occurs when the net is driven by two or more pins, regardless of their logic values. For example, two drivers with 1R and 1S states would be in contention. 1S and 0S would also be in contention.
- Same. Contention occurs when the net is driven by two or more pins that have the same logic values. For example, two drivers on the same net with 0S and 0S states would be in contention. The states 1R and 0S would not be in contention. This model is useful for open collector circuitry.
- **Different**. Contention occurs when the net is driven by two or more pins with opposite logic values. For example, two drivers with 1S and 0R states would be in contention. States 1R and 1S would not be in contention.
- **Driven**. This is a custom property that allows you to specify your own contention conditions. The examples in the dialog box show how to construct the contention model. For example, a Driven value of (?? & 0s) would be in contention when a 0s was present with any other state.

You can assign the Contention property to any bus or net, and you can specify these items using selection or specific design names. When you assign the property to a bus or net, the property affects only that bus or net. When assigning the Contention property, you can also specify an amount of time that the contention condition must exist before the simulator generates any warning message.

## **Conditions That Affect Kernel Setup**

Condition	Setup Item	Description
Design contains zero-delay feedback loops.	Iteration limit	Set iteration limits for initialization and run time.
VHDL models assertions set.	HDL assertion severity level	Set the level where VHDL assertions stop simulator.
VHDL models have signals specified as arrays.	HDL array size	Set array elements that \$examine_objects() shows.
You are debugging design logic.	Timing mode	Select the unit timing mode.
You are debugging the effects of timing on logic.	Timing mode	Select the minimum, typical, or maximum timing mode.
You are debugging design constraints.	Timing mode	Select timing   constraint mode, enable checking.
Design contains nets with multiple drivers.	Net Contention	Enable contention checking.
Check the simulation for spike conditions.	Spike model and spike check	Select spike model and enable spike checking.
Design contains QuickParts that generate messages.	Quickpart messages	Enable the display of QuickPart messages.
Check technology files for unspecified timing paths.	Unspecified paths check	Enable unspecified path checking.
Check for hazard conditions.	Hazard check	Enable hazard checking.
Check for thoroughness of pin toggling.	Toggle check	Enable pin toggle checking.

## **Conditions That Affect Kernel Setup**

Before setting up the kernel, you should consider the contents of your design and where you are in the flow of simulation development. The table contains some typical conditions that call for setting up the kernel before the simulation.

## Lab Preview

- Enable full constraint checking
- Set the X-immediate spike model and enable spike checking
- Use Change Warning Start to remove initialization warnings.
- Use cross-window highlighting to find the source instance and net in warning messages
- Use stimulus timing adjustments to optimize timing with constraint limits
- Examine the entire simulation process

#### Lab Preview

In the lab exercise for this module, you will:

- Enable full constraint checking by providing all the invocation switches to the quicksim shell command.
- Set the X-immediate spike model and enable spike checking using the quicksim shell invocations options.
- Use Change Warning Start to remove initialization warnings at the beginning of the simulation run.
- Use cross-window highlighting to find the source instance and net in warning messages
- Use stimulus timing adjustments to optimize timing with constraint limits
- Examine the entire simulation process.

### Lab Exercise



If you are reading this workbook online, you might want to print out the lab exercises to have them handy when you are at your workstation.

#### **Procedure 1: Setting Up the Debug Environment**

In this lab procedure, you will enable full constraint checking so that QuickSim II will determine if your design is operating with allowed constraints. You are to use a "canned" stimulus file that will produce constraint violations. Your mission is to remove as many of the violations as possible.

- 1. If necessary, log into your workstation and set your current directory to your qsim\_n directory.
- 2. Invoke QuickSim II on the design viewpoint for add\_det using shell invocation, enabling all of the constraint checks, as follows:

your\_path/training/qsim\_n
shell> \$MGC\_HOME/bin/quicksim add\_det/default
-tim typ -consm messages -spm x\_immediate
-spw all -hac on -coc on -mm on

This viewpoint contains the design changes that you made in Module 10.

- 3. Close the Info Messages window (the 🔔 stroke).
- 4. Open the root sheet (the \_\_\_\_\_ stroke).

- 5. Verify that the kernel is set up for proper signal checking by doing the following steps:
  - a. Choose: (Menu bar) > <u>Setup > Kernel</u>
  - b. Click on: Visible

The display shows you the current kernel conditions that were set when you invoked QuickSim II using the switch options, as shown:

Setup An	alysis		
Timing mode     Current     Unit     D       Details of 'Current' timing mode     Hi	elay Constraint dden Visible		
Timing mode =typChangeConstraint modeOffState onlySpike modelX immediateSuppress	Delay Scale     1     Override       Messages     Override       s     Override		
Image: Second structure       Image: Second structure       Spike warnings to display:         Image: Second structure       Image: Second structure       Spike warnings to display:         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure         Image: Second structure       Image: Second structure       Image: Second structure			
OK Reset Cancel			

c. Use the  $\checkmark$  stroke to cancel the form.

6. Report object information on the 74ls161a instance (the r stroke):

The Objects window appears. This window shows the values for the kernel setup condition on this instance. These values should be the same as for the entire design

Dbjects	L
<pre>(previous text not shown) Simulation object data: Model type : QP Spike model : x_immediate Constraint mode : messages Timing mode : typ Delay scale : 1 Checks - hazard : On spike reports contention : On toggle unspecified path : Off stability Model messages : On stability state Total pins</pre>	: All : Off : Off : OFF

7. Close the Object report window (the  $\downarrow$  stroke).

- 8. Change the timing mode and spike check on the 74ls161a instance as follows:
  - a. Choose: **Setup > Kernel > Change > Timing mode**
  - b. Fill out the dialog box as follows:

Change Timing Mode		
On Selected instances Named instances		
Timing Mode =minDelay Scale1.2	Change	
OK Reset Ca	ancel Help	

- c. Choose: Setup > Kernel > Change > Spike Warnings
- d. Fill out the dialog box as follows:

Change Spike Warnings		
On Selected instances	Named instances	
Spike warnings to display: Suppress Transport X Display All Warnings	Override	
OK Reset Cancel Help		

e. Report on the 74ls161a instance to verify that changes were made.

9. Load a forcefile as follows:

To allow you to debug timing and constraint problems, a forcefile named forcefile\_demo has been created beneath the add\_det container. You will load and use this stimulus to debug the design.

- a. Choose: (Menu bar) > Setup > Force > From File
- b. Use the Navigator button to locate and select the *forcefile\_demo* object beneath the *add\_det* object.
- c. **OK** the navigator and the dialog box.

This file loads into the forces WDB.

- 10. Next you will start a clock on the PULSE signal, as follows
  - a. First, select the PULSE signal.
  - b. Click on: **STIMULUS**



- c. Set the clock period to 100nsec, and the duty cycle (the percentage of the clock period that the clock is a 1) to 50%. Start the clock at the 0 state (Clock is active: "High").
- d. **OK** the dialog box.

- 11. Build the Trace, List, and Monitor windows as follows:
  - a. Maximize the schematic view so you can select the signals to add to the following windows.
  - b. Using the Trace  $\neg$  and List  $\downarrow$  strokes, add all input and output signals to the Trace and List windows.
  - c. Now add only the output nets to the Monitor window (from the Debug Gates palette).
  - d. Use the **MGC** > **Cleanup Windows** menu item to position all windows in their default positions and sizes.
- 12. Initialize the entire design to "0". (Hint: Use the Init command.)

#### **Procedure 2: Fixing Constraint Violations**

1. Run the simulator for 3600nsec using the run command.

The simulator runs and stops at time 3600 as shown in the message area.

A Simulation Messages window appears telling you that you had many violations in the simulation. This report appeared because you enabled messages in the setup. Some of the messages are shown here for convenience:

Simulation Messages		
Spike at time 0.0ns on net '/I\$245/N\$218'.		$\square$
Spike at time 0.0ns on net '/I\$245/N\$237'.		
Spike at time 0.0ns on net '/I\$245/N\$220'.		
Spike at time 0.0ns on net '/N\$277'.		
Spike at time 0.0ns on net '/FULL'.		
Setup time violation at time 9.0ns, on instance '/I\$7'		
Simulated: 9.0ns, Specified: 15.0ns		
On pin 'D' To pin '_E'.		
Setup time violation at time 9.0ns, on instance '/I\$8'		
Simulated: 9.0ns, Specified: 15.0ns		
On pin 'D' To pin '_E'.		
Hold time violation at time 12.0ns, on instance '/I\$8'		
Simulated: 3.0ns, Specified: 20.0ns		
On pin 'AO' To pin '_E'.		
Hold time violation at time 12.0ns, on instance '/I\$8'		
Simulated: 3.0ns, Specified: 20.0ns		
On pin 'Al' To pin '_E'.		
Hold time violation at time 12.0ns, on instance '/I\$8'		
Simulated: 3.0ns, Specified: 20.0ns		
On pin 'A2' To pin '_E'.		
Hold time violation at time 12.0ns, on instance '/I\$7'		
Simulated: 3.0ns, Specified: 20.0ns		
On pin 'AO' To pin '_E'.		
Hold time violation at time 12.0ns, on instance '/I\$7'		
Simulated: 3.0ns, Specified: 20.0ns		
On pin 'Al' To pin '_E'.		
Hold time violation at time 12.0ns, on instance '/I\$7'		
Simulated: 3.0ns, Specified: 20.0ns	<u> </u>	
	$\geq$	

- 2. Oops! We forgot to turn off warning messages during the initialization part of the run (spikes reported at time 0.0).
  - a. Reset the simulator.
  - b. Change the Warning Start delay to 5 nsec

Enable Messages and/or memory invalidations at time 5

- c. Initialize the circuit to "0".
- d. Run again for 3600 nsec.

You will notice that now the spike messages at time 0.0 are suppressed. Three setup and six hold violations are still reported.

- 3. Grow the Simulation Messages window and select by handle as follows:
  - a. Click on the Maximize button for the Simulation Messages window.
  - b. Now identify the instance '/I\$7' with the spike by unselecting all objects, and selecting the text in the Simulation Messages window.
  - c. Pop the schematic view window to view it. Use the (Popup) > View > Selected menu item if the selected instance is not in view. This instance is the right 74259 instance.
  - d. Identify the instance '/I\$8' using the same procedure.

- 4. Determine the source of the setup and hold problems, as follows:
  - a. Unselect everything and select the " E" text in the Simulation Messages window for the '/I\$7' instance.

The E pin on the second 74259 selects. Why didn't the E pin on the first 74259 instance also select?

b. Click on: DBG GATES > Nets The net connected to the \_E pin is selected. This net comes from the inverter gate with a typ delay of 9 nsec. Therefore, this setup problem is also due to initialize state at time 0.0 propagating through the inverter 9 nsec later. The Warning Start time did not account for all delays.

Note that the 'E' pin is also the source of the hold problem. The change on this pin at 9 nsec is too soon for A0-A2 to be stable.

5. Use the **Report > Timing** menu item to write down the Technology file statements that generate these setup and hold messages:

tS = \_\_\_\_\_ tH = \_\_\_\_\_

Do NOT close the Timing Info report window.

- 6. Fix the setup and hold problems by adding custom init properties to the nets connected to the '\_E' pins, as follows:
  - a. Select the net connected to the '\_E' pin on the left 74259 instance.
  - b. Click on the Add Property palette icon in the Design Changes palette.
  - c. Enter the following and OK the Add Property dialog box: New Property Name: init Property Value: 1 Property Type: String (Do not choose "Number")

Note that the Timing Info report window is invalidated (lined out) to indicate that a timing/state change has been made. In this case, the information is still valid, since init values do not affect Technology file source.

d. Using steps a-c above, add the same init property to the net connected to the '\_E' pin on the right 74259 instance.

Next you will verify this fix.

- 7. Reset the simulation to time 0.
- 8. Initialize the design to 0 again.

This initializes the entire design, except for the nets connected to '\_E' pins, which will be initialized to 1. List the '\_E' pins to verify this.

9. Run the simulation again for 3600.

The Simulation Messages window shows that the six Hold violations still exist. This is because they trigger on the '(AH)' transition on the '\_E' pin. Since '\_E' goes high at time zero due to the init value of "1", the violations still occur. Next you will try a different approach to fix the problem.

- 10. Fix the timing problem and check the result, as follows:
  - a. Delete the Init property on each nets connected to the '\_E' pins.
  - b. Add the Init property to the input to each of the inverters that outputs to the '\_E' pins.
  - c. Reset the simulation.
  - d. Initialize the design to 0.
  - e. Run for 3600 nsec.

Notice that only one Setup violation remains. This is not an initialization problem, but a true circuit timing problem.

	Simulation Messages		
Se	etup time violation at time 250.0ns, on instance '/I\$9	1	
	On pin '_LOAD' To pin 'CLK'		
		$\triangleright$	

11. Analyze the problem and formulate a solution.

What signal drives the '\_LOAD' pin? \_\_\_\_\_

What signal drives the '\_CLK' pin? \_\_\_\_\_

It appears that the start pulse is the problem. How far must we shift the START pulse to avoid the setup problem?

Another alternative is to create the clock on /PULSE again, this time starting it at the 1 state so that at time 250 it has a negative going (non-clocking) edge.

- 12. Change the /PULSE clock, as follows:
  - a. Reset the simulation.
  - b. Reissue the /PULSE clock signal as you did in Procedure 1, Step 10. This time chose the "Clock is Active: **Low**"
  - c. Initialize the circuit to 0.
  - d. Run for 3600 nsec.

This didn't work. It just changed the violation to time 200 where the leading edge of the /START pulse occurs.

- 13. Change the /START pulse, as follows:
  - a. Reissue the /PULSE clock signal, as you did in Procedure 1, Step 10. The clock should begin in the 0 state (active high).
  - b. Reset the simulation.
  - c. Use the waveform editor to move the start pulse 25 nsec earlier. (Hint: setup the waveform editor for a 25 nsec grid spacing and use the Shift palette button.)
  - d. Initialize the design to 0.
  - e. Run the simulation for 3600 nsec.

Finally, all simulation messages have been eliminated.

Use the techniques you have learned in this workbook to verify that no unknowns remain in the design, that the counter is working properly, and that the ACCESS signal goes to FFFF at 3025.8nsec.

14. Exit QuickSim II.

- 15. Final notes on the entire simulation process:
  - Setup the environment and kernel first.
  - Initialize your circuit, since a simulation reset will apply default initialization values in the circuit again.
  - Set the Warning Start time so that no initialization messages display.
  - Run the simulator until the circuit stabilizes (fully initialized).
  - Issue your stimulus, remembering that relative stimulus will start at the current simulation time, while absolute stimulus (includes all waveform databases) will assume absolute time 0 as reference.
  - Run the simulation.
  - Save any states, Trace or List window data, results, or status reports before resetting or exiting, since this information will be lost.



#### LAB EXERCISE COMPLETED

Complete the Test Your Knowledge questions on the next page and compare your answers to those in "Module 11 Solutions" on page A-19.

Then continue on to the next module.

#### **Test Your Knowledge**

Answer these questions and compare your answers to those in "Module 11 Solutions" on page A-19.

1. What is the default spike model in QuickSim II?

How does this model treat spikes?

- 2. What is the difference between a Probe and a Monitor Flag.
- 3. What operation do you normally perform just prior to an "Unselect Except X" operation.
- 4. What does the "Driven" contention model all you to do?

## **Module Summary**

In this module, you enabled kernel checking to determine if your design was operating within the limits specified by the models. The following types of checking can be performed in the QuickSim II simulation kernel

- Spike model and spike check. Spikes are pulses that are shorter than the input to output delay of the model.
- Hazard. An instance is evaluated more than once in the same timestep due to zero-delay feedback loops within the design. Hazards cannot occur with unit-delay timing.
- Constraint checks. These are specified in the Technology file and determine if timing is within the following limits:
  - Setup. Arrives prior to a defined signal
  - Hold. Does not change for a period of time after a signal is stable.
  - Width. Pulse does not occur shorter than a defined value
  - Fmin/Fmax. The frequency of a signal is within limits.

The elimination of all timing constraints is the final part of a design simulation process. This process includes these three steps:

- Check design functionality using unit-delay timing.
- Check design functionality using typical timing values. You can optionally use min and max timing values to verify proper operation under marginal timing conditions.
- Check design timing with constraints. This verifies that all instances in the design are operating within allowed limits.

# Appendix A Solutions

This appendix contains all of the answers to the Lab Exercises and the Test Your Knowledge questions.

#### **Module 1 Solutions**

#### Lab Answers

Step 8c: What is the name of the icon created? Transcript

Step 9c: What is the time displayed in the Monitor Window? 0.0ns

#### **Test Your Knowledge Solutions**

- 1. Name the common simulation user interface: SimView/UI
- 2. True or False. It is mandatory to have a design viewpoint to use QuickSim II on the design? **True**
- 3. What two applications automatically create required design objects when QuickSim II is invoked?1. Design Viewpoint Editor
  - $2. \ {\rm TimeBase}$
- 4. Name the two parts of an electronic design.
  - 1. Component (Design)
  - 2. Configuration (Viewpoint)

- 5. What Run options are available with the [palette] Run button? For Time, Until Time, Until Stop, and Resume
- To stop the simulation on your workstation type during a run, use the \_\_\_\_\_\_
   Ctrl-C in a UNIX shell and Ctrl-S in the Aegis shell \_\_\_\_ key.
- 7. When you exit QuickSim II, what information are you prompted to save: Waveform Data, Simulation Setup, and Design Changes

## **Module 2 Solutions**

#### Lab Answers

#### Procedure 1.

Step 5: What are the session default waveform databases?

- Default Waveform Database: \_\_results\_\_
- Force Target Waveform Database: \_\_\_forces\_\_\_

Step 8c: Using the icons listed in the Notepad navigator dialog box as identifiers, match the files with the appropriate waveform file format:



(Draw a line to connect the file with its format type)

Step 11b: A Trace window is displayed containing two waveforms. What are the names of the two waveforms shown in the Trace window?

1. \_\_forces@@/CLR\_\_

2. \_\_forces@@/PRE\_\_

#### Procedure 2.

Step 18: What are the current values of /Q and /QB?

#### **Test Your Knowledge Solutions**

1. What form must stimulus take before it is issued to the kernel?

#### \_\_\_Waveform database form\_\_\_

- 2. What are the benefits of storing waveform information in binary form vs. ASCII form?
  - Waveform database issues stimulus to the kernel fast
  - The stimulus can be previewed in the Trace and List windows
  - Graphical stimulus editing is available
- 3. What two ASCII formats can waveform database output be converted to?
  - a. \_\_logfile format\_\_
  - b. \_\_forcefile format\_\_
- 4. When you issue the Force command, what waveform database is the force entered into, by default? \_\_forces (or Force Target)\_\_
- 5. What menu item do you use to load a forcefile into a waveform database?

\_\_Setup > Forces > From File\_\_
## **Module 3 Solutions**

## Lab Answers

#### Procedure 1

Step 3e: List the path here: your\_path/training/qsim\_n/LATCH/default\_1

Step 6: Function Name = **\$add\_primitive()** 

Step 7e: When does this rule take effect? The next evaluation...QuickSim II invocation

#### Procedure 2

- Step 2: 1. Timing mode: minimum
  - 2. Constraint mode: off
  - 3. Simulator resolution: **1.0 nsec**
  - 4. Model messages: on
  - 5. Delay scale: 1
  - 6. Spike model: suppress
- Step 6d: i. Simulation Model property value: \$G5
  - ii. Simulation model type: **QP**
  - iii. 74ls161a symbol type: MG\_STD
  - iv. 74ls161a component version: 1
  - v. Simulation timing mode: min
  - vi. Spike checking: on
  - vii. Hazard checking: off
  - viii. State of the "B" input pin: **0s**
  - ix. State of the "RCO" output pin: Xs

- 1. Which QuickSim II invoke switch loads a setup object? -setup
- 2. Define "design viewpoint": The design configuration rules, and back annotations used by your design

- 3. If you have a property name that has no value, what DVE operations can you use to assign a value to that property? Add Parameter
- 4. What is the name of the default viewpoint created by QuickSim II or DVE if you do not specify a name? **default**
- 5. List the DVE menu path that configures the viewpoint for QuickSim II. Setup > (Quick)SIM, Fault, Path and Grade
- 6. How do you perform a design check within DVE? Miscellaneous > Check Design > Check Options

## **Module 4 Solutions**

### Lab Answers

#### Procedure 1.

- Step 3: What type of functional model do you suppose this component uses, by default? **schematic**
- Step 4: What type of functional simulation model do you suppose this component uses, by default? **QPT or schematic? You can't tell.**
- Step 5: Which one is the default symbol? You can't tell.Which functional model is used for simulation?None. Rip is for connectivity only.
- Step 9b: What line number describes the behavior of the dff component when the CLR signal is unknown? 21What are the outputs under this condition? X, X

- 1. Which symbol property provides a path to RAM or ROM initialization files? (Circle one) Modelfile
- 2. Define component registration: Assigning a label, type and model path in the component interface.
- 3. The value of the \_\_model\_\_ property is matched against existing component interface labels to determine model selection.
- 4. Overloading refers to the process of creating your own models for the \_\_\_\_\_ builtin primitive\_\_\_ models.
- 5. Which of the following are supported in QuickSim II? (circle all that apply).
  - a. QuickPart Table models
  - b. QuickPart use of primitive switches

#### c. QuickPart hierarchical models

- 6. True or False: Behavioral Language Models (BLMs) can use technology files for timing and technology information. **True**
- 7. True or False: VHDL models and other modeling techniques cannot be mixed in the same schematic design. **False**

## **Module 5 Solutions**

### Lab Answers

#### Procedure 1:

- 4. What are the current settings of the Setup Environment dialog box?
  - i. User Scale Type: \_\_time\_\_ Scale: \_\_1e-09\_\_
  - ii. Write Report Page Width: \_\_80\_\_ Length: \_\_66\_\_
  - iii. Exit queries \_\_\_Design, Setup, WDBs, and Timing\_\_\_
  - iv. Default force type \_\_Charge\_\_
  - v. Default display radix \_\_Hex\_\_
- 20. Why are there only 4 X's for the 16-bit ACCESS signal? Each X represents 4 bits of information since the default display radix is hex.

#### **Procedure 3:**

- 3. What operation does the "!" symbol perform? \_\_Boolean NOT\_\_
- 5. What happened to expression 4? <u>\_\_\_lt was removed\_\_</u> Why? <u>\_\_</u> The expression is no longer valid\_\_\_\_
- 6. Did expression 4 return? \_\_\_No, you must recreate it\_\_\_
- 11a. Using the List window information, at what time does expr1 go true 2600.1 What time does expr2 go true 3350.2
  - b. Why are there so many transitions on the expr3 trace? QuickSim II shows a transition every time the expression is evaluated.
  - c. What boolean value must these two have to make expr4 true? (FULL=1s and access(15)=0) OR (FULL!=1s AND access=1)

- Which of the following setups can't you report on using a Report menu item?
   Current Working Directory
- 2. Where do you set the Window Layout mode? In the Session setup
- 3. What is the default name of a saved QuickSim II setup object? \_\_\_\_\_quicksim\_setup\_\_\_
- 4. Keeping circuit activity is a function of QuickSim II operating with SimView/UI. A keep list stores signal names and their event transitions in which waveform database?\_\_results\_\_
- 5. Where do you set the default force type? In the environment setup
- 6. How would you globally scale all timing values? Change the "Delay scale" value in the kernel setup

## **Module 6 Solutions**

## Lab Answers

#### Procedure 1

- Step 2: Timing mode: unit delay Delay scale: 1 (no scaling) Simulator resolution: 0.1 nsec
- Step 5: What is the current state of all the nets in the design? XrWhat is the current simulation time? 0.0 nsecWhat type of initialization was performed? default
- Step 8c: Does the simulation stop due to an "all quiet" condition? Yes Do any of the signals remain at X? Yes Which ones? All of the outputs
- Step 9d: Does the simulation stop due to an "all quiet" condition? **Yes** Do any of the signals remain at X? **No**

#### **Procedure 2**

Step 7: Which net name corresponds to bit 0 of the bus? N\$1

- Three drivers 1S, 0R, and XI are all driving the same net. What is the resultant state on the net? 1S & 0R = 1S & XI = XS
- 2. By default, which type of initialization does QuickSim II perform upon invocation? **Default (no design stabilization run is performed**
- 3. Why is it important to run the simulation after initialization but before issuing stimulus? This stabilizes your design by propagating the initial states throughout the design

- 4. Describe how a bus appears in the Trace window for each of the following:
  - 1. Combined \_\_as a single signal with the new name\_\_
  - 2. NOCombined \_\_as the individual signals\_\_

## **Module 7 Solutions**

### Lab Answers

#### Procedure 1:

5d. What are the names and values of the six waveforms shown in the Trace window?

Waveform Name	Initial Value
1forces@@/TEST	Xr
2forces@@/PULSE	Xr
3forces@@/ANALOG_OUT	Xr
4forces@@/LATCH	Xr
5forces@@/START	1s
6forces@@/_CLR	1s

#### **Procedure 2: Working with Waveform Databases**

1a. A new Trace window appears. What is the name of this window?
 \_\_Trace#2\_\_\_

- 1. When you issue the Force command, what waveform database is the force entered into, by default? \_\_forces (or Force Target)\_\_
- 2. When you use the Waveform Editor palette, what waveform database default mechanism determines the waveform database into which the edits are entered? \_\_ Force Target\_\_

- What palette icon would you use to change the initial event value for a new waveform about to be created? \_\_ [Waveform Editor] Setup Waveform Editor\_\_
- 4. In terms of a QuickSim II menu item, palette icon or command, what does "make a waveform database persistent" mean?\_\_**Save WDB**\_\_
- 5. What is the menu path for online help information about the individual palette icons? \_\_Help > On Palettes > Palette Descriptions\_\_

## **Module 8 Solutions**

### Lab Answers

#### Procedure 1:

- Step 6: What do you think will happen at the input to the first inverter (I1) when you run the simulation? It should "see" the 1r and evaluate the change
- Step 7: When the simulation stops, what are the current signal states?

OUT1 \_X\_ OUT2 \_X\_ OUT3 \_X\_

Step 8c: Did the circuit begin clocking? YES

What is the period (time of one cycle) of the outputs? **0.1 nsec** 

Why? The kernel is currently using unit delay timing

Step 9c: Before you run the simulation again, what are the delays for each of the inverters? **zero** 

What will happen in the circuit? The circuit will oscillate

Step 10d: What will happen in the circuit on the next run. No oscillation

- 1. What three categories determine the timing accuracy in a digital simulation?
  - 1. simulator timing resolution
  - 2. timing model attributes (min/typ/max, constraints)
  - 3. special routines (pin loading, process data)

- What is the difference between an iteration and an event?
   Iteration -- a complete evaluation of all models once. Also, part of a timestep.
   Event -- a state change for a signal at a given simulation time
- 3. What conditions can occur when a signal enters a zero-delay feedback loop?
  - a. an Oscillation Limit Error occurs
  - b. the event queue is cleared for that loop
- List the two modes of resetting the simulation state and describe the difference.
   Reset command
   Change a model
- 5. How does an actionpoint differ from a breakpoint?
  - 1. A breakpoint stops the simulator on the condition
  - 2. An actionpoint pauses the simulation, executes the actionlist, and resumes the simulation
- 6. What are the steps you use to perform a batch simulation?1. create a set of stimulus (normally in waveform db form
  - 2. create a batch file of setup conditions
  - 3. run the simulation, invoking displayless with redirected batch file
  - 4. examine the results waveform database in SimView
- 7. Which menu item do you use to view the current breakpoints.(Menu bar) > Report > Setup > Breakpoints
- 8. Describe the function of the breakpoint action list. An AMPLE list of commands that get executed when the break occurs
- 9. When does a breakpoint occur if you set "On Occurrence" = 0x10? It occurs on the 16th occurrence of the condition.

## **Module 9 Solutions**

### Lab Answers

11e. What are the new waveform's names?

forces@@/N\$1	forces@@/N\$480
forces@@/N\$479	forces@@/N\$8

## **Test Your Knowledge Solutions**

- 1. What does a red dot in a traced waveform indicate?\_\_a redundant (or repeated) waveform event\_\_
- 2. Assertion tests are a result of comparing waveforms from what two waveform databases?
  - asserts waveform database (represent expected results)
  - results waveform database (represent actual results)
- 3. What feature might you use to identify the exact time that a traced edge occurs?



NEAREST

## **Module 10 Solutions**

### Lab Answers

Step 11a: What are the three field headings it contains?

- 1. Instance Pathname:
  - 2. Net Pathnames:
  - 3. Pin Pathnames

What type of changes appear beneath the "Pin Pathname" heading? Changes to the Rise and Fall properties

Step 13f: Which entry heading does the new change appear beneath? **"Net Pathname" heading** 

What is the value for this property?

10\*tcap because there is no value assigned to TCAP. If TCAP were defined the value would be a real number.

- 1. List two ways to change a model in QuickSim II:
  - $1. \ \mbox{Use}$  the Change Model operation and choose from the model list.
  - 2. Use the Change Property operation to change the value of the model property.
- 2. What determines the back annotation object priority when more than one BAO is connected? The most recently connected BAO has the highest priority
- 3. Name the benefits that the design viewpoint provides:
  - 1. Provide the configuration rules for the design.
  - 2. Is a container in which you can place other saved objects.

## **Module 11 Solutions**

## Lab Answers

- Step 4a: Why didn't the \_E pin on the first 74259 instance also select? The \_E is really a hierarchical path that is associated only with the object that produced it, in this case, /l\$7.
- Step 11: What signal drives the '\_LOAD' pin? /START What signal drives the '\_CLK' pin? /PULSE How far must we shift the START pulse to avoid the setup problem? At least 24 nsec

## **Test Your Knowledge Solutions**

1. What is the default spike model in QuickSim II? Suppress

How does this model treat spikes? The pulse event is not scheduled.

- What is the difference between a Probe and a Monitor Flag.
   A Probe is a synonym for a net name or handle...it does not display waveform or state information. A Monitor flag is a graphical object placed on a net or pin that contains state information
- 3. What operation do you normally perform just prior to an "Unselect Except X" operation. Select All -nets
- 4. What does the "Driven" contention model all you to do?It allows you to precisely define the contention condition.

# Appendix B Design Manager Icons

This section summarizes the Idea Station design object types. Object types are identified by the Design Manager based on fileset extensions and are then displayed as unique icons.

## Data Object Icons

The following table lists the navigator window icons registered with the Design Manager. This table is also included in the *Design Manger User's Manual*. The column titled "V" specifies whether the object is versioned where the version is indicated as "#".

Design Object	Icon	Purpose of Design Object	Fileset (X is object name)	V
Eddm_ba_mgr	ba	Back annotation data object.	default.Eddm_ba_mgr.attr default.ba_#	Y
Eddm_design_ viewpoint	dvpt	Design viewpoint. A container object that defines the design configuration rules and references back annotation objects.	default (directory) default.Eddm_design_ viewpoint.attr default.dvpt_#	Y
Eddm_part		Object ties together all models associated with a component.	part.Eddm_part.attr part.part_#	Y

Table B-1. D	esign Objec	t Navigation	Icons
--------------	-------------	--------------	-------

Design Object	$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{i$				
Design Object	Icon	Purpose of Design Object	Fileset (X is object name)	V	
Hdl_arch_do		VHDL architecture. Compiled functional description. Default: X= "behav"	X.Hdl_arch_do.attr X.arch_# X.lif_# X.sav_# X.sobj_# X.sym_#	Y	
Hdl_entity_do		VHDL entity. Compiled interface description. Default: X= "entity"	X.Hdl_entity_do.attr X.ent_# X.lif_# X.sav_# X.sobj_# X.sym_#	Y	
Hdl_pkg_body_ do	B	Compiled VHDL package body object. Default: X= "body"	X.Hdl_pkg_body_do.att X.hob_# X.lif_# X.sav_# X.sobj_# X.sym_#	Y	
Hdl_pkg_hdr_ do		Compiled VHDL package header object. Default: X= "header"	X.Hdl_pkg_hdr_do.attr X.hdr_# X.lif_# X.sav_# X.sobj_# X.sym_#	Y	
Mgc_blm	BLM	Behavioral Language Model.	X.mgc_blm.attr		
Mgc_component		Component. A container object.	X (directory) X.mgc_component.attr	N	
Mgc_container		A directory or directory with attribute file.	X (directory) X.mgc_container.attr	N	

Table B-1. Design Object Navigation Icons

Design Object	Icon	Purpose of Design Object	Fileset (X is object name)	V
Mgc_schematic		Schematic. Container for sheets. References and is referenced by mgc_sheet objects.	schematic (directory) schematic.mgc_ schematic.attr schematic/schem_id	N
Mgc_sheet		Sheet. Contains graphical and connectivity data for Design Architect sheets.	sheet1.mgc_sheet.attr sheet1.sgfx_# sheet1.ssht_#	Y
Mgc_symbol	μ Π	Symbol. Contains graphical symbol and properties.	X.mgc_symbol.attr X.smbl_#	Y
Qpb_g5_model	QPS	Compiled QuickPart Schematic model.	X.Qpb_g5_model.attr X.qp_#	Y
Tdm_qpt_do	QPT	Compiled QuickPart Table model.	X.Tdm_qpt_do.attr X.qpt_#	Y
Simv_kernel_ setup		Database created when saving setup conditions. Default: X= <b>app_name</b> _setup Example: "quicksim_setup"	X (directory) X.Simv_kernel_setup.attr	N
Simv_log	LOG	ASCII file containing logfile information used in QuickSim.	X.log	N
Simv_misl	MISL	ASCII misl file.	X.misl	N

Table	B-1.	Design	Object	Navigation	Icons

Design Object	Icon	Purpose of Design Object	Fileset (X is object name)	V
Simv_save_state		Database created when saving QuickSim II state information.	quicksim_state (directory) quicksim_state.Simv_ save_state.attr	N
Svdm_svdb		Simulation vector data model. Waveform database files used with simulation.	X.Svdm_svdb.attr X.dat_# X.wdb_#	N
Tf_tfile_do	TECH	Compiled technology file. Default: X= "technology"	X.Tf_tfile_do.attr X.tecf_#	Y
vhdl		VHDL source file that is edited in Design Architect.	X.vhdl.attr X.vhdl_#	Y

Table B-1. Design Object Navigation Icons

## **Idea Station Tool Icons**

The Design Manager examines a design structure and compares design objects to registered tool viewpoints. A tool viewpoint contains a tool icon for that tool. Table B-2 lists the tool icons used with Idea Station designs.

Application Name	<b>Tool Icon</b>	<b>Tool Icon Name</b>
Design Architect		design_arch
Design Viewpoint Editor	C) @ Dzb dvpt	DVE

 Table B-2. Idea Station Application Tool Icons

Application Name	<b>Tool Icon</b>	Tool Icon Name
Notepad		editor
QuickFault II	<b>□</b> <sup> </sup>	QuickFaultII
QuickGrade II	<u>□<b>२</b></u> ₽-₽	QuickGradeII
QuickPath	<mark>ے پُون</mark> ٹو طنیق	QuickPath
QuickSim II	<u>০</u> ০ শৃহন্দ	QuickSimII
QuickSim II-FPGA	<u>고 Q</u> 뜻2~-	QuickSimII-FPGA
SimView (stand-alone)		SimView
System 1076	xu[2] sys_1076	sys_1076

Table B-2. Idea Station Application Tool Icons